appviewx

Automation Guides

Version: 2024.1.0.0

Copyright AppViewX, Inc.

Copyright © 2025 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.

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface 15

Revision History 15

About the Automation Guides 15

Audience 15

Third-Party Software Acknowledgements 15

Text Conventions 15

Chapter 1. Automation User Guide 17

Module Overview 17

Solutions 17

Salient Features 18

Accessing the Automation Module 19

Prerequisites 19

Web Browser Requirement 19

Getting Started with Automation 20

Automation Concepts 20

Configuring Role Based Access Control 21

Resource 21

Role 23

User Group 27

User 28

Getting Started with Prebuilt Workflow Tasks 28

Accessing FAQs 29

Building Auto-generate Forms 30

Sample Workflows using Prebuilt Tasks 31

Application Delivery Automation 47

Automate DNS Services 66

Change Automation 72

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

DevOps and Continuous Configuration Automation 74
GitOps Integration 77
Utilities 78
SDK Automation 80
Workflow Studio 81
Accessing the Workflow Page 82
Workflow Actions 85
Designing a Workflow 92
Accessing Solutions 94
Accessing the Store 97
Importing Workflows 104
Collection 105
Hooks Inventory 113
Magic Variable 125
Helper Scripts 130
Regex Library 138
Request Settings 141
Provisioning for Workflow Tasks 142
Source Control 143
Integration 158
Workflow Tasks 216
Accessing Workflow Tasks 218
Task Category - General 219
Task Category - User Interface 275
Task Category - ChatOps and Notifications 360
Task Category - Change Management 379
Provisioning for Workflow Tasks 380
Connecting Workflow Tasks 382
Adding Variables to Tasks 386

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Automation Collision 389
Workflow Task Actions 391
Rollback Workflow 399
Variable Mapping 403
Scheduling Tasks 413
Workflow Cart 415
Version Control 418
Workflow Options 421
Northbound Integrations 432
Service Requests 455
Configuring RBAC for Workflow Request Stage View 456
Accessing Service Requests 457
Request :: Overview 457
Custom Reports 459
Request :: View/Run 460
Scheduled Jobs 467
Request :: All 468
Request :: Open 468
Request :: Closed 468
Request :: Failed 469
Request :: Assigned 469
Request :: Audit Logs 469
Configuring RBAC for Archive and Restore Requests 470
Archive Requests 471
Restore Archives 473
Preferences 474
Service Request Actions 478
Automation and Service Catalog 487
..... 487

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Chapter 2. Automation API Guide....489
API Reference....489
Enable ACF Permission for API Reference....489
Best Practices for Working with the AppViewX API....490
Understanding the AppViewX Automation API....491
RESTful HTTPS Requests....491
Requests....492
Request Structure....493
Response Structure....493
Description of Server Responses....493
URI Scheme....494
Types of Accounts in AppViewX....494
Authentication Using a User Account....494
Retrieve session ID using login API....495
Using Session ID for further API calls....500
Authentication Using a Service Account....504
Retrieve Access Token using get-service-token API....504
Using Access Token in the header for further API calls....508
Submit Request....512
Before you begin....512
Request Structure....513
Task Specific Information....514
Response Structure....524
Status Codes....525
Reference....526
Update Request....527
Before you begin....527
Request Structure....528
Payload....529

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Task Specific Information..... 530
Grid Task Payload..... 535
Response Structure..... 537
Status Codes..... 538
Reference..... 538
Get Current Task Detail of the Request..... 539
Before you begin..... 539
Request Structure..... 540
Payload..... 541
Response Structure..... 542
Status Codes..... 542
Sample Request/Response..... 543
Reference..... 546
Get Request Summary/Log..... 547
Before you begin..... 547
Request Structure..... 548
Response Structure..... 549
Status Codes..... 550
Sample Request/Response..... 550
Reference..... 552
Get Associated Workorder..... 553
Before you begin..... 553
Request Structure..... 554
Response Structure..... 555
Status Codes..... 555
Sample Request/Response..... 556
Reference..... 557
Get Tasks which are in InProgress State..... 557
Before you begin..... 558

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Structure..... 558
Payload..... 559
Response Structure..... 560
Status Codes..... 560
Sample Request/Response..... 561
Reference..... 562
Upload File..... 563
Before you begin..... 563
Request Structure..... 563
Payload..... 565
Response Structure..... 566
Status Codes..... 566
Sample Request/Response..... 567
Reference..... 569
Download File..... 569
Before you begin..... 569
Request Structure..... 570
Payload..... 571
Response Structure..... 572
Status Codes..... 572
Sample Request/Response..... 573
Reference..... 573
Delete File..... 574
Before you begin..... 574
Request Structure..... 575
Payload..... 576
Response Structure..... 576
Status Codes..... 577
Sample Request/Response..... 578

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Reference....578
Abort Request....579
Before you begin....579
Request Structure....580
Response Structure....581
Status Codes....582
Sample Request/Response....582
Reference....582
Clone Request....583
Before you begin....583
Request Structure....584
Response Structure....585
Status Codes....586
Sample Request/Response....586
Reference....587
Get All Requests....588
Before you begin....588
Request Structure....589
Response Structure....590
Status Codes....591
Sample Request/Response....591
Reference....597
Pause Request....598
Before you begin....598
Request Structure....599
Response Structure....600
Status Codes....601
Sample Request/Response....601
Reference....602

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Resume Request ... 603
Before you begin ... 603
Request Structure ... 603
Response Structure ... 605
Status Codes ... 605
Sample Request/Response ... 606
Reference ... 606
Generate Commands ... 607
Request Structure ... 607
Response Structure ... 608
Status Codes ... 609
Sample Request/Response ... 610
Reference ... 614
Generate Commands And Implement ... 615
Request Structure ... 615
Response Structure ... 616
Status Codes ... 617
Sample Request/Response ... 618
Reference ... 621
Rollback Request ... 621
Before you begin ... 621
Request Structure ... 622
Response Structure ... 623
Status Codes ... 624
Sample Request/Response ... 624
Reference ... 625
Execute Hook ... 626
Before you begin ... 626
Request Structure ... 627

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure..... 628
Status Codes..... 629
Sample Request/Response..... 629
Reference..... 630
Collection Operations..... 631
Before you begin..... 631
Request Structure..... 632
Payload..... 633
Response Structure..... 634
Status Codes..... 635
Sample Request/Response..... 635
Sample Request/Response..... 636
Sample Request/Response..... 637
Sample Request/Response..... 638
Sample Request/Response..... 639
Sample Request/Response..... 641
Reference..... 642
Get Collection Details..... 643
Before you begin..... 643
Request Structure..... 644
Payload..... 645
Response Structure..... 646
Status Codes..... 646
Sample Request/Response..... 647
Reference..... 648
Collection Exists in DB..... 648
Before you begin..... 648
Request Structure..... 649
Payload..... 650

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure..... 651
Status Codes..... 651
Sample Request/Response..... 651
Reference..... 652
Update Workflow Resource Permission..... 653
Before you begin..... 653
Request Structure..... 653
Payload..... 654
Response Structure..... 654
Status Codes..... 655
Sample Request/Response..... 656
Reference..... 657
Chapter 3. Automation Troubleshooting Guide..... 659
Prerequisites..... 659
Issues with Workflows..... 659
Issues in Creating a Workflow..... 660
Issues in Importing a Workflow..... 660
Issues in Exporting a Workflow..... 661
Issues in Workflow Validation..... 661
Issues with Workflow Tasks..... 662
Issues in Adding Tasks 662
Issues in Updating Tasks..... 663
Issues in User Interface Tasks..... 663
Issues in Using Helper Scripts..... 664
Issues in Creating a Helper Script..... 664
Issues in Updating a Helper Script..... 664
Issues in Importing a Helper Script..... 665
Issues in Using Hooks..... 665
Issues in Creating a Query explorer Hook..... 665

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Issues in Creating a Script Hook....666
Issues in Creating a REST API Hook....666
Issues in Exporting Hooks....666
Issues in Importing Hooks....667
Issues in Creating a Workflow Request....667
Issues in Submitting a User Task....667
Issues in Submitting an Executed Service Request....668
Issues in Reflection of Workflow Data Updates in a Service Request....668
Issues in the Form task in a Service Request....668
Issues in Data Manipulation in a Service Request....669
Issues in Command Task in a Service Request....669
Chapter 4. Service Catalog User Guide....670
Module Overview....670
Prerequisites....672
Getting Started with the Service Catalog....672
Accessing the Service Catalog....673
Design....673
Published....675
Store....677
Import....677
Getting Started with OOB Service Catalog....679
Using OOB Catalogs....680
Take a Tour....681
Customizing Prebuilt Service Catalogs....682
Designing a Custom Service Catalog....682
Getting Started with Designing a Custom Catalog....683
Page Layout....684
Build....703
Reports....863

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Workflow....874
Folders.... 880
Inventory....886
How to Undo/Redo actions....896
Moving widgets.... 897
Cloning Widgets....898
Version Control.... 899
Preview a Service Catalog.... 902
Publishing a Service Catalog....902
Sharing a Service Catalog....903
Sharing a Service Catalog with a User(s)....903
Sharing a Service Catalog with a User group(s)....904
Scenarios.... 905
Service Catalogs with Birthright Role 906
Limitation.... 906
Cloning a Service Catalog....906
Cloning a Published Catalog....906
Cloning a Catalog in Design mode.... 907
Setting up the Landing Page....908
Setting up the Landing Page when Publishing/Sharing.... 908
Setting up the Landing Page under User Preferences....909
Hiding the Page Header 910
Troubleshooting....912

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Preface

Revision History

Revision	Description	Date
1.0	Initial release of document for Release 2024.1.0.0	March 2025

About the Automation Guides

This section includes the following guides that give you an overview of the functionalities offered in the Automation module:

- Automation User Guide
- Automation API Guide
- Automation Troubleshooting Guide
- Service Catalog User Guide

Audience

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.

Third-Party Software Acknowledgements

This section serves as a placeholder to document the third-party components referenced in this guide, along with their associated trademark information.

For example,

- This document includes software details developed by VMware, Inc. (www.vmware.com).

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Convention	Description
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands with a paragraph, URLs, codes in examples, text that appears on the screen, or text that you enter.
	Indicates a feature available only in the on-premise deployments of AppViewX.
	Indicates a feature available only in the SaaS deployments of AppViewX.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Chapter 1: Automation User Guide

Welcome to the complete guide to getting started with Automation. Learn how to use workflows for Network Service Orchestration and Automation with AppViewX modules, including working with inventory, connecting tasks and apps, and designing workflows to automate repetitive tasks and solutions.

- Module Overview
- Prerequisites
- Getting Started with Automation
- Configuring Role Based Access Control
- Getting Started with Prebuilt Workflow Tasks
- Workflow Studio
- Workflow Tasks
- Service Requests
- Automation and Service Catalog

Module Overview

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

The module 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. Its 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 Automation module 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)

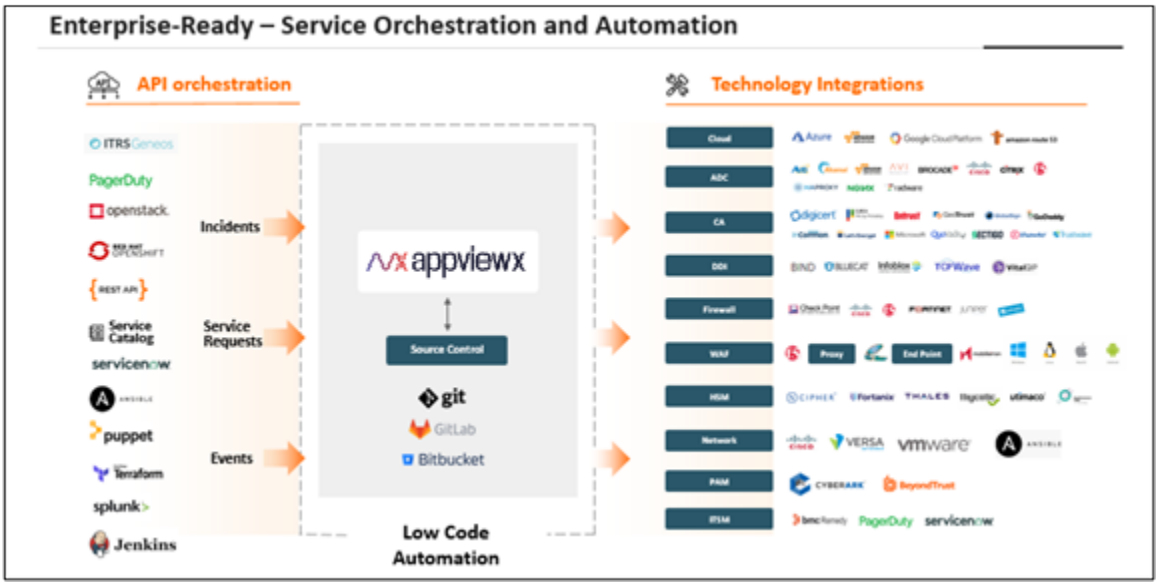
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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
- **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 © 2025 AppViewX, Inc. All Rights Reserved.



• Accessing the Automation Module

Accessing the Automation Module

1. Hover your mouse over the **(Menu)** icon.
2. From the menu displayed, select **Automation**.
   The Automation module is displayed with the **Request :: Overview** section open by default. There are three sections here:
   - **Service Requests**: The Service Requests section allows you to execute workflow(s). Workflows can be executed manually or can be scheduled.
   - **Workflow Studio**: The Workflow Studio allows you to design simple and complex workflow(s) from scratch and also use the out of the box (OOB) workflows.
   - **Service Catalog**: The Service Catalog allows you to design and customize self-service catalogs by persona.

Prerequisites

Web Browser Requirement

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Browsers	Version
Google Chrome	v85.0.4183.83 (64-bit) or later

Getting Started with Automation

Automation Concepts

- **Workflow Studio**: An inventory to design and manage automation workflows and design self-service catalogs for NetOps, SecOps, App teams and PKI admin.

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

- **Tasks**: Tasks are the units of action (aka automation lego blocks) in workflows. 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.

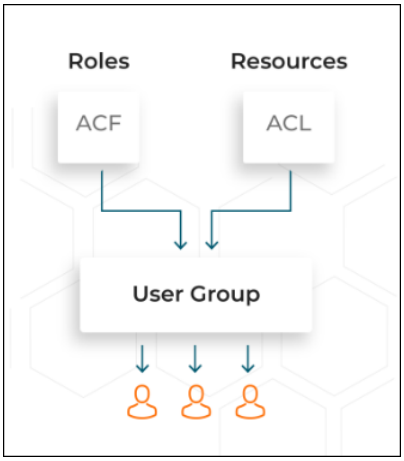
- **Service Requests**: The Service Requests section allows you to execute workflow(s). Workflows can be executed manually or can be scheduled.

  For more information, refer to the section on Service Requests.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring Role Based Access Control

AppViewX offers comprehensive support for Role and Resource-Based Access Control (RBAC). RBAC is a method of restricting AppViewX functions, network resources that can be managed and monitored in AppViewX based on the roles of individual users within an enterprise. It allows you to integrate with the existing identity stores such as Active Directory (AD) and Lightweight Directory Access Protocol (LDAP) to enforce authorization policies. Roles and Resources can be customized to suit any organizational structure and user requirements.



RBAC lets employees have access rights only to the AppViewX functions and network resources they need to do their jobs and prevents them from accessing information that doesn't pertain to them.

- Resource
- Role
- User Group
- User

Resource

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

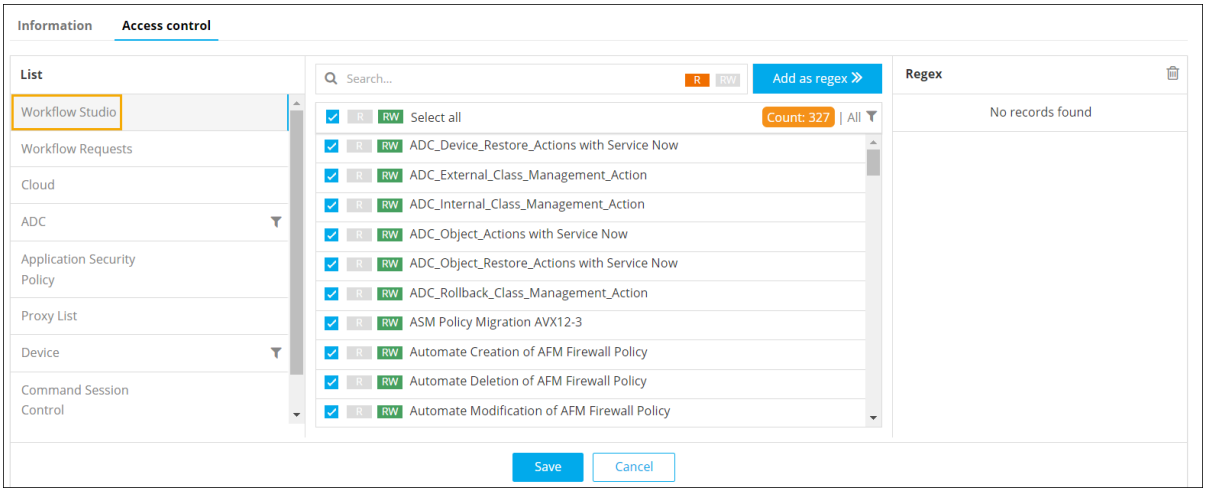
New Menu	Old Menu
In the **Platform** module, from the left pane, under **IDENTITY**, select **Resource**. The **Resource** page is displayed, with all the roles listed with their **Name**, **Description**, and **Status**.	From the main menu, select **Account** > **Resource**. The **Resource** page is displayed, with all the roles listed with their **Name**, **Description**, and **Status**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

New Menu	Old Menu

Note: For more information on how to switch between menus, click here.

1. On the **Resource** page, click on any resource.
2. To see the permissions that can be defined for the resource in the Workflow Studio, under the **Access Control** tab, from the **List** menu, select **Workflow Studio**.

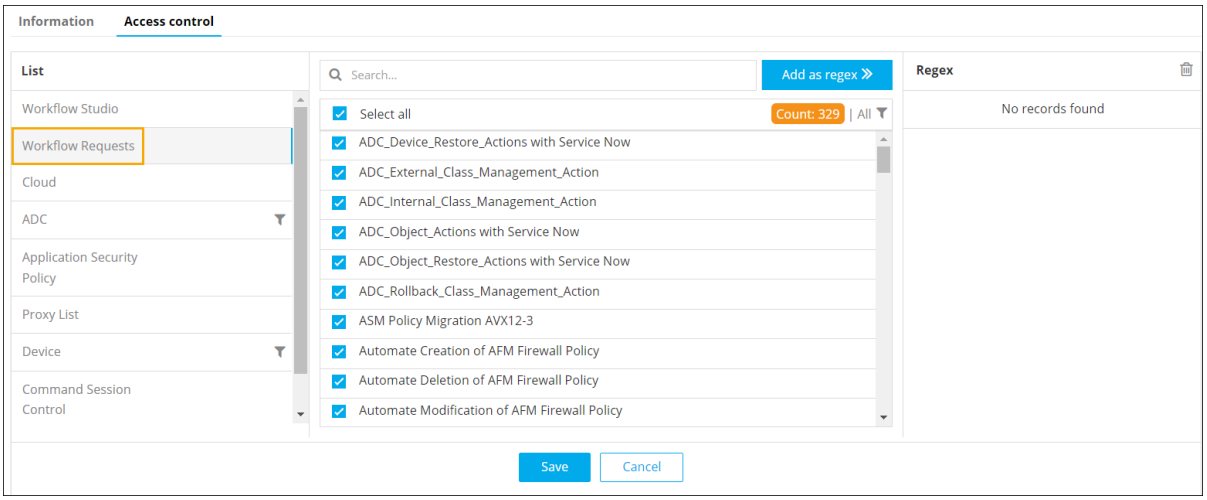


The following permission(s) can be defined for the resource:

- 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/Deselect: If you deselect a particular workflow, that workflow will not be available on the Workflow page for this resource.
- When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. To see the permissions that can be defined for the resource in the Service Requests console, select **Workflow Requests**.



- When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.

Note: If you deselect a particular workflow here, it will not be available on the Request :: View/Run page for this resource.

4. Click **Save**.

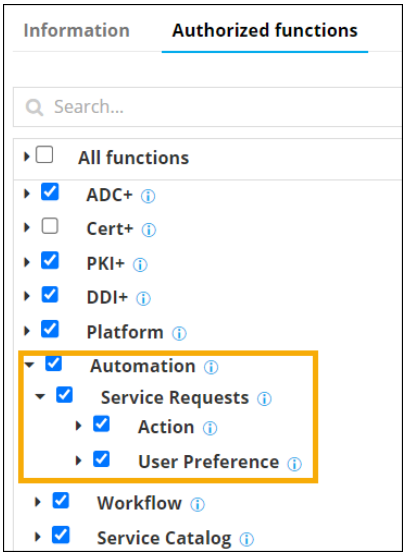
Role

To get an overview of the actions authorized to a role:

New Menu	Old Menu
In the **Platform** module, from the left pane, under **IDENTITY**, select **Role**. The **Role** page is displayed, with all the roles listed with their **Name**, **Description**, and **Status**.	From the main menu, select **Account** > **Role**. The **Role** page is displayed, with all the roles listed with their **Name**, **Description**, and **Status** .
Note: For more information on how to switch between menus, click here.	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. On the **Role** page, select any role from the list of roles, for example **admin**.
2. To see the list of permissions that can be defined for this role, under the **Authorized Functions** tab, expand **Automation** > **Service Requests**.



Note: If you are using the old menu, under the **Authorized Functions** tab, expand **Request** > **Workflow**.

3. To see the permissions available under **Action**, click **(Select Right)** icon.

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.
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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Action	Description
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.

4. Select/Deselect the checkbox to allow/restrict that action for the selected user.
5. To see the permissions available under **User Preference**, click **(Select Right)** icon.

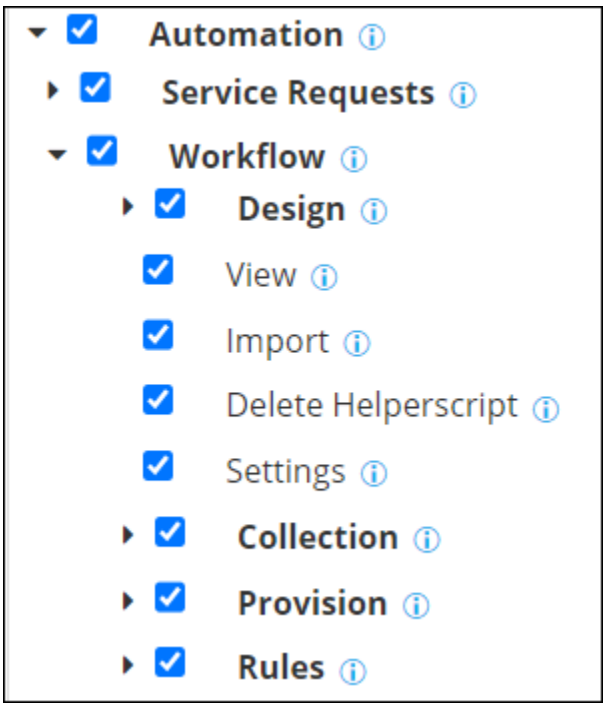
Various actions that can be authorized or restricted for a user under User Preference

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Action	Description
Show Request stage	Provision to show or hide the service request stage view for a user.

6. Select/Deselect the checkbox to allow/restrict that action for the selected user.
7. To see the permissions that can be defined for the user in the Workflow Studio, expand **Automation** > **Workflow**.



Note: If you are using the old menu, under the **Authorized Functions** tab, expand **Request** > **Workflow**.

Various actions that can be authorised or restricted for a user in the Workflow Studio module

Action	Description
Design	Provision to allow or restrict a user to design a workflow. The following permissions are available under Design: • **Create/Modify**: Create a new workflow or modify an existing workflow • **Delete**: Delete existing workflow(s). • **Clone**: Clone existing workflow(s).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Action	Description
	• **Export**: Export workflows (that are in disabled state) to other environments. • **Shared folder**: Share folder(s) containing logically grouped workflows.
View	Provision to allow or restrict a user to view detailed information about a workflow.
Import	Provision to allow or restrict a user to import workflows from other environments.
Delete Helperscript	Provision to allow or restrict a user to delete helper script(s).
Settings	Provision to allow or restrict a user to update workflow settings.
Collection	Provision to allow or restrict a user to access Collection.
Provision	Provision to allow or restrict a user to manage provisioning settings.
Rules	Provision to allow or restrict a user to define action-based rules to trigger workflows for ADC objects and dynamically tag resources.

8. Select/Deselect the checkbox to allow/restrict that action for the selected user.
9. Once the required permissions have been assigned, click **Save**.

User Group

You can see the details of a particular User Group such as the Roles and Resources mapped to it.

New Menu	Old Menu
In the **Platform** module, from the left pane, under **IDENTITY**, select **Usergroup**. The **User Group**page is displayed, with all the user groups listed with their **Name**, **Description**, and other related information.	From the main menu, select **Account** > **Usergroup**. The **User Group**page is displayed, with all the user groups listed with their **Name**, **Description**, and other related information.
Note: For more information on how to switch between menus, click here.	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

User

You can see the details of a group that a User is mapped to and also modify user information.

New Menu	Old Menu
In the **Platform** module, from the left pane, under **IDENTITY**, select **User**. The **User** page is displayed, with all the user groups listed with their **Name**, **Full Name**, and other related information.	From the main menu, select **Account** > **User**. The **User** page is displayed, with all the user groups listed with their **Name**, **Description**, and other related information.
Note: For more information on how to switch between menus, click here.	

Getting Started with Prebuilt Workflow Tasks

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.

Note: You can also access FAQs and a few Pre-built workflows to get you started in the Solutions section.

- Accessing FAQs
- Building 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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Accessing FAQs

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.

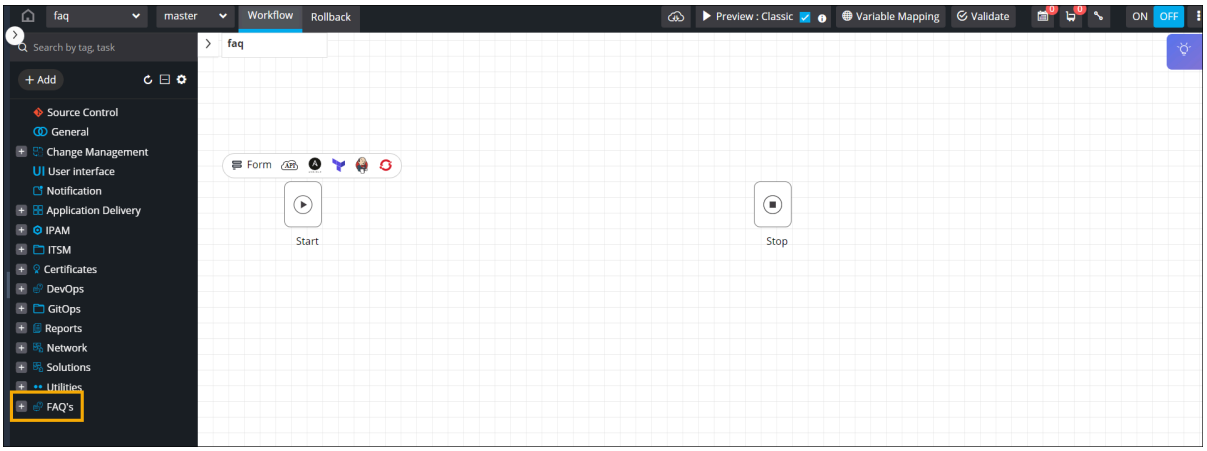
Following are a few sample flows that will help you learn how to connect tasks:

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

To access the FAQs folder from the workflow design space:

1. Go to **Automation > WORKFLOW > Design** the **Workflow** page is displayed.
2. On the **Workflow** page that is displayed, click **Create New**.

   The design space opens, with the FAQs folder visible in the left menu.



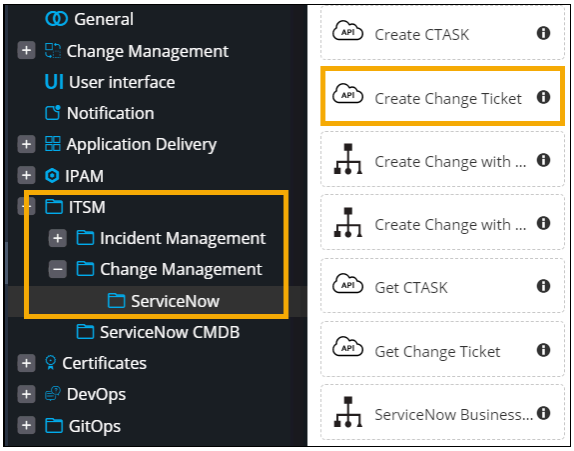
Note: You can also access the FAQs by opening any of the existing workflows on the **Workflow** page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

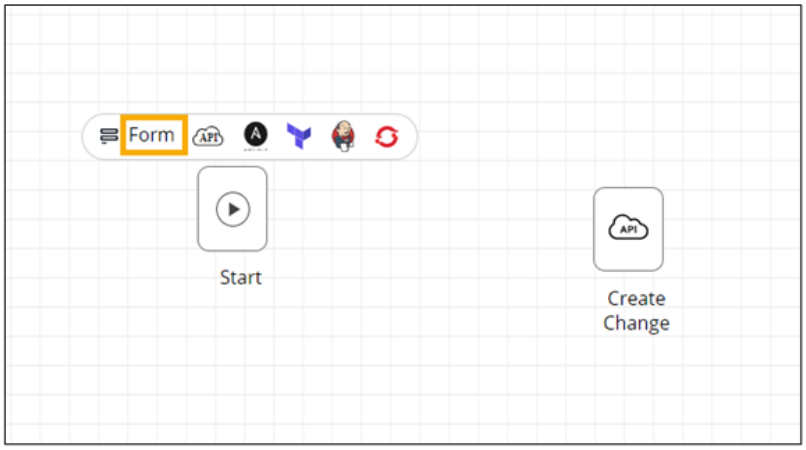
Building Auto-generate Forms

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. Design a new workflow.
2. Drag and drop a prebuilt workflow task. For example, **Create Change Ticket** task from the **ITSM** > **Change Management** > **ServiceNow** folder.

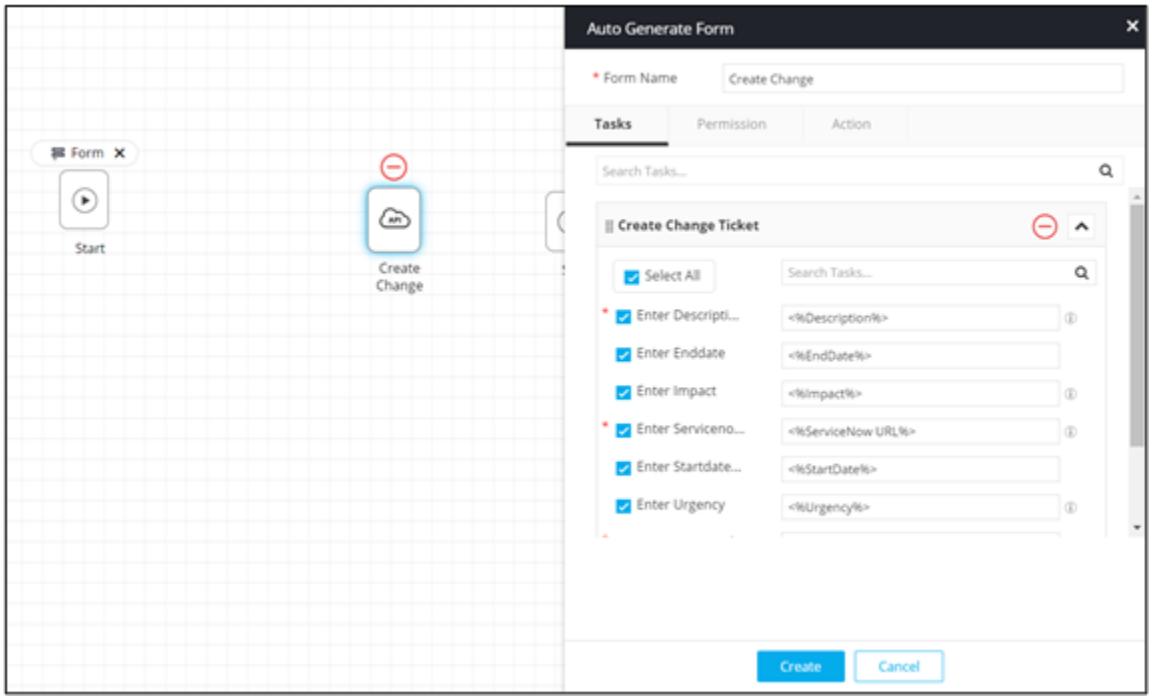


3. To auto-generate a self-service form, click **Form** above the **Start** task.



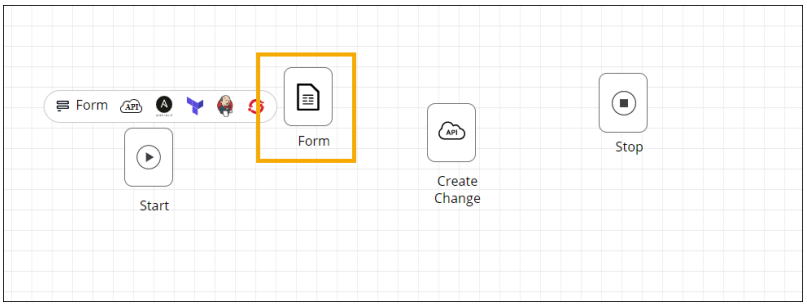
   The **Auto Generate Form** window is displayed.
4. Enter the **Form Name**.
5. To auto-fill the variables in the form, click **(Add)** icon above the **Create Change** task.
   The input tasks for the **Create Change** task are auto-populated in the **Auto Generate Form** window, under **Tasks**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Select/deselect the fields to be displayed in the self-service form and click **Create**.

   The **Form** task is created and added to the design space.



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.

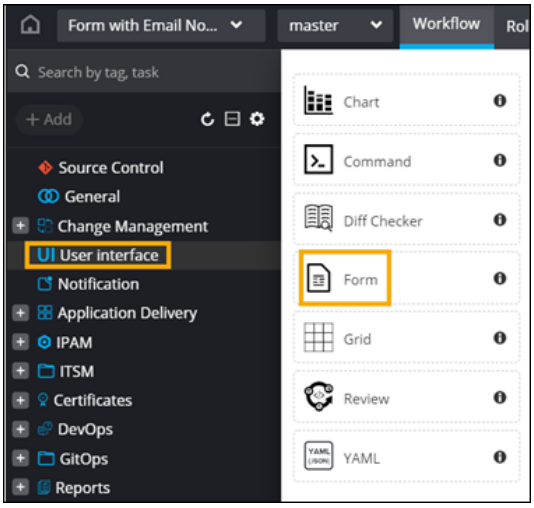
- Form with Email Notification
- Executing a Ping Check using Command Task
- Creating A Record on Infoblox device using prebuilt tasks
- Creating a VIP with Incident Ticket

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form with Email Notification

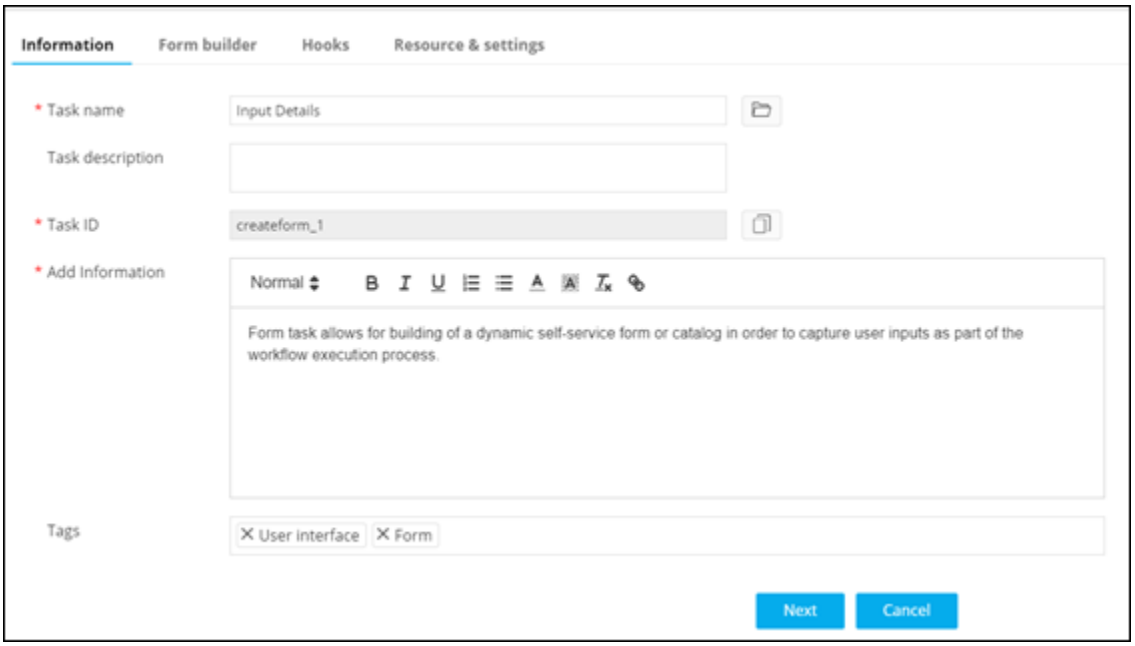
You can design a workflow to send an email notification based on the details provided in a form.

1. Design a new workflow.
2. In the design workspace, from the User Interface section in the left menu, drag and drop a **Form** task.



The **Gallery** window is displayed on the screen.

3. In the **Gallery** window, to design a new form, click **(Create Form)** icon.
4. In the Form task window that is displayed, under the **Information** tab, enter the **Task name**.



5. Click **Next**.

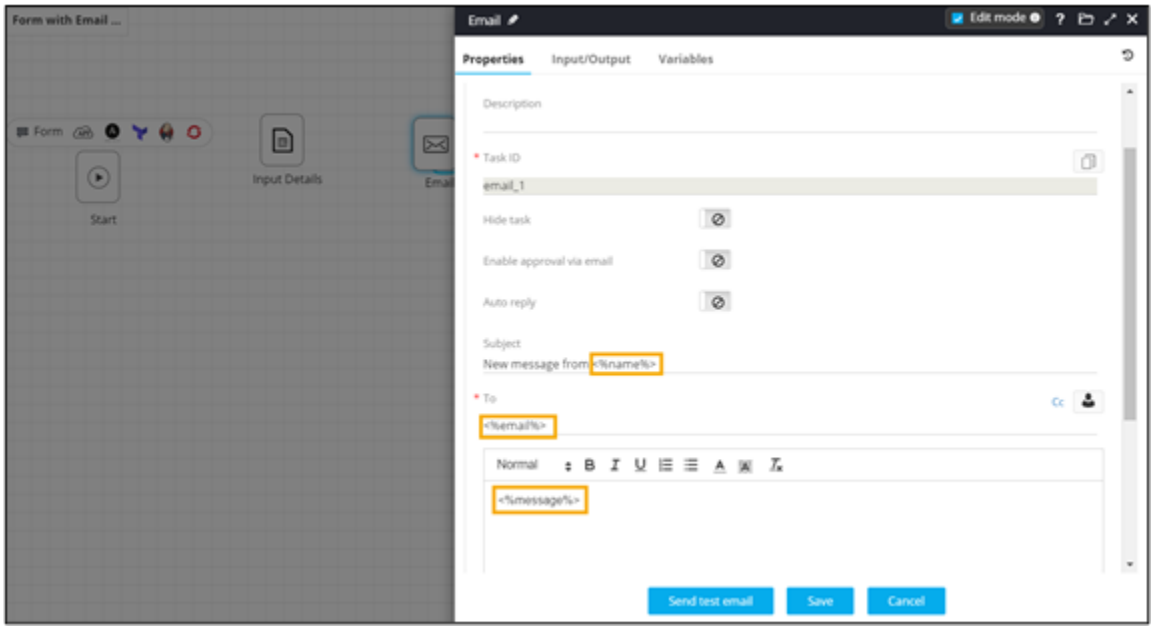
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The **Form builder** tab opens with the **Field Properties** window displayed at the bottom right corner of the screen.

6. Add the following form fields and turn on the Global variable toggle to enable the field IDs as global variables:

Label name	Field type	Field ID
Enter your name	Text box	name
Enter your email address	Text box	email
Enter your message	Multi-line	message

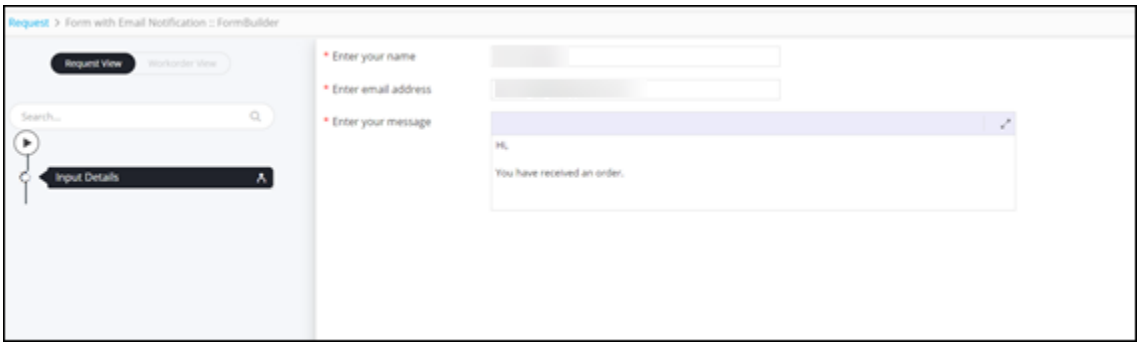
7. Click **Save**.
8. From the Notification section in the left menu, drag and drop **Email** task.
9. In the **Email** task window that is displayed, under **Properties**, in the **General** section, refer the global variables defined in the Form task.



10. Click **Save**.
11. Connect the workflow tasks and enable the workflow.
12. Trigger the workflow.

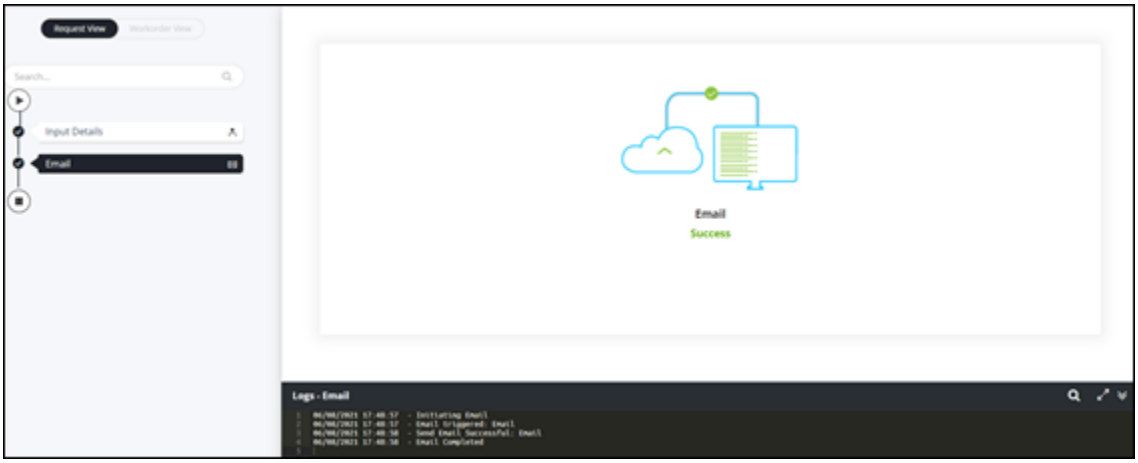
    The workflow execution page is displayed with the workflow inputs requested at the first stage.
13. Enter the details in the input form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



14. Click **Submit**.

    Email notification is successfully sent to the email address provided in the form.

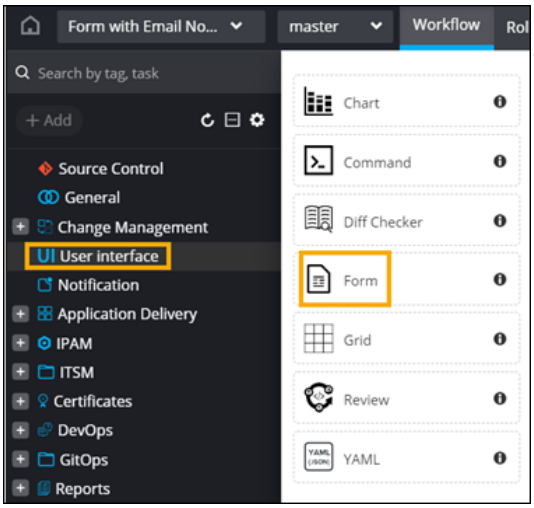


Executing a Ping Check using Command Task

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

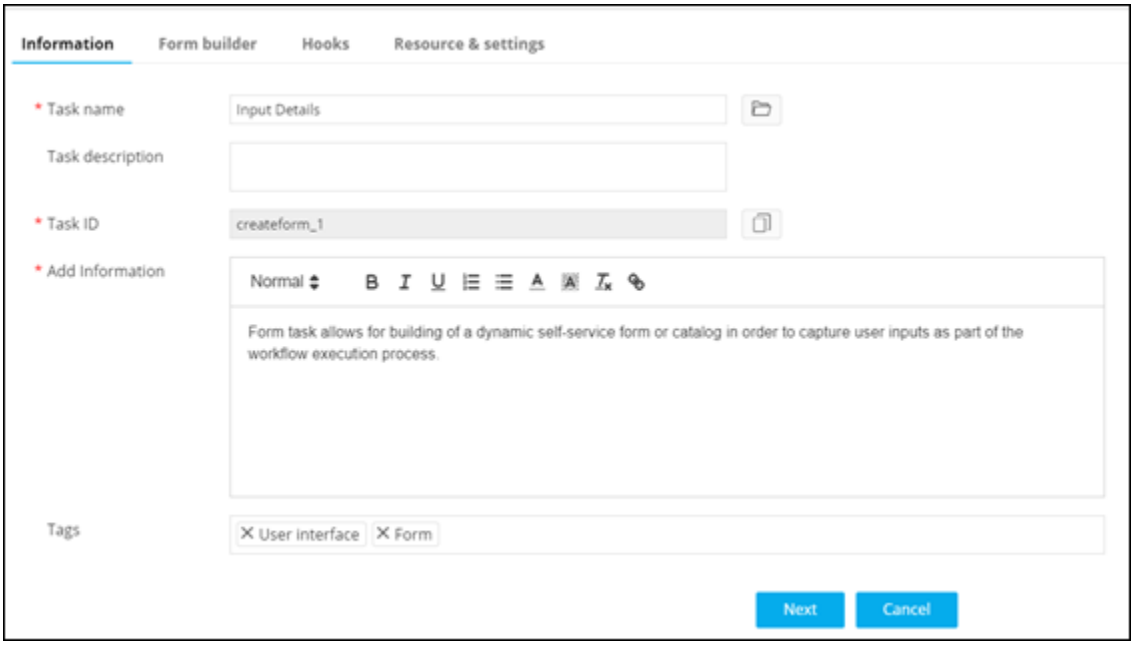
1. Design a new workflow.
2. In the design workspace, from the **User Interface** section in the left menu, drag and drop a **Form** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **Gallery** window is displayed on the screen.

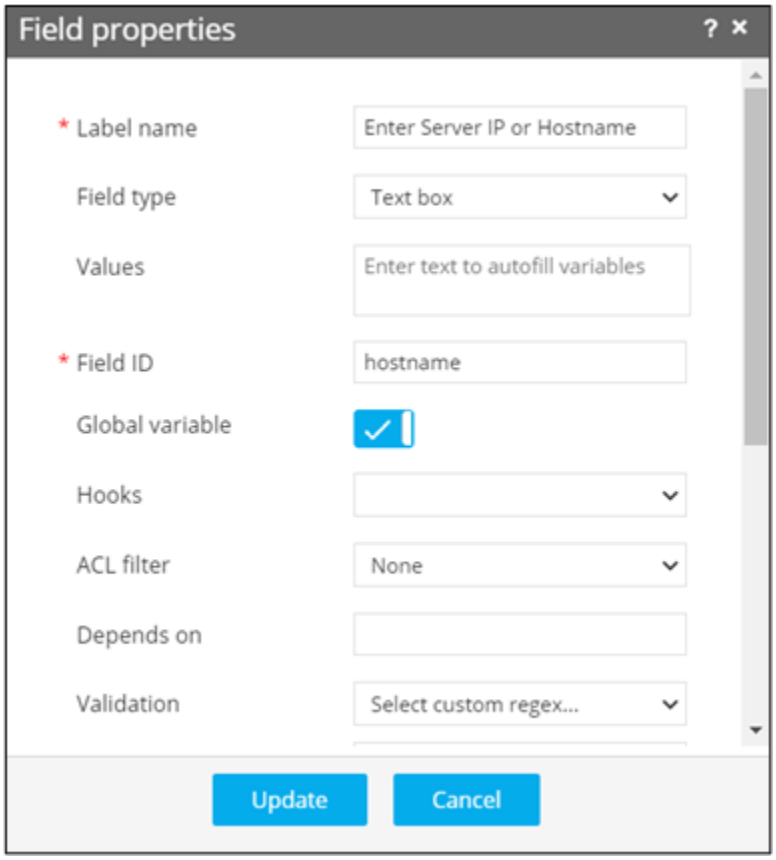
3. In the **Gallery** window, to design a new form, click **(Create Form)** icon.
4. In the **Form** task window that is displayed, under the **Information** tab, enter the **Task name**.



5. Click **Next**.

   The **Form builder** tab opens with the **Field Properties** window displayed at the bottom right corner of the screen.
6. In the **Field properties** window, define the form field to get the Server IP or Hostname as shown.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

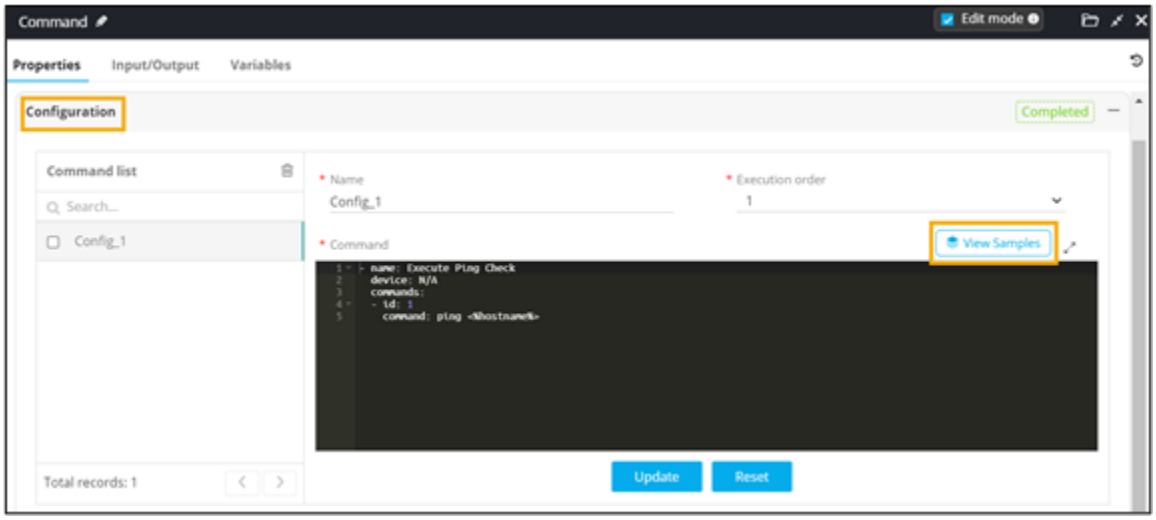


7. To declare the Field ID as a global variable, `<%hostname%>`, and refer it in the next task, turn on the **Global variable** toggle.

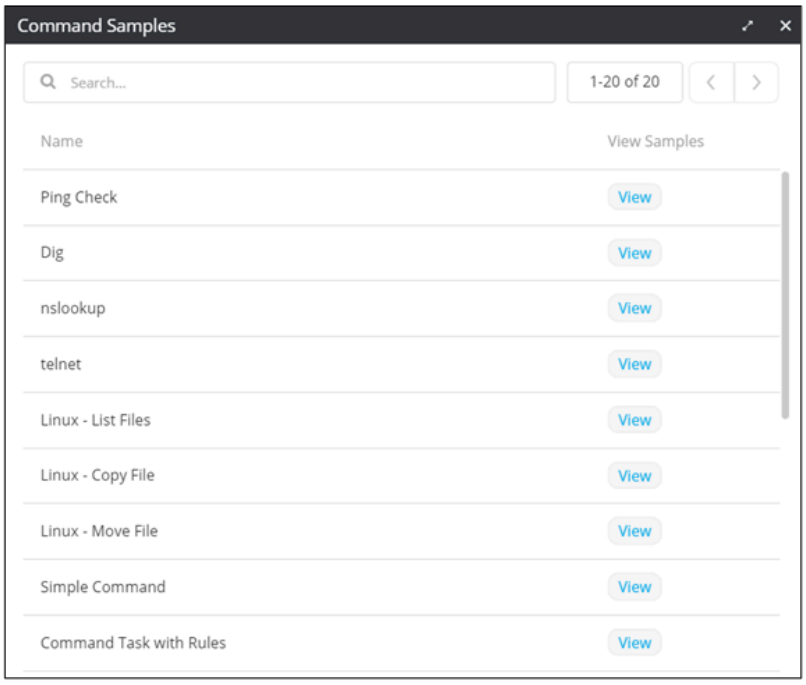
> **Note:** For more information on adding form fields, click here.

8. Click **Save**.
9. From the User Interface section in the left menu, drag and drop the **Command** task.
10. In the **Command** task window, under **Properties**, in the **Configuration** section, click **View Samples**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

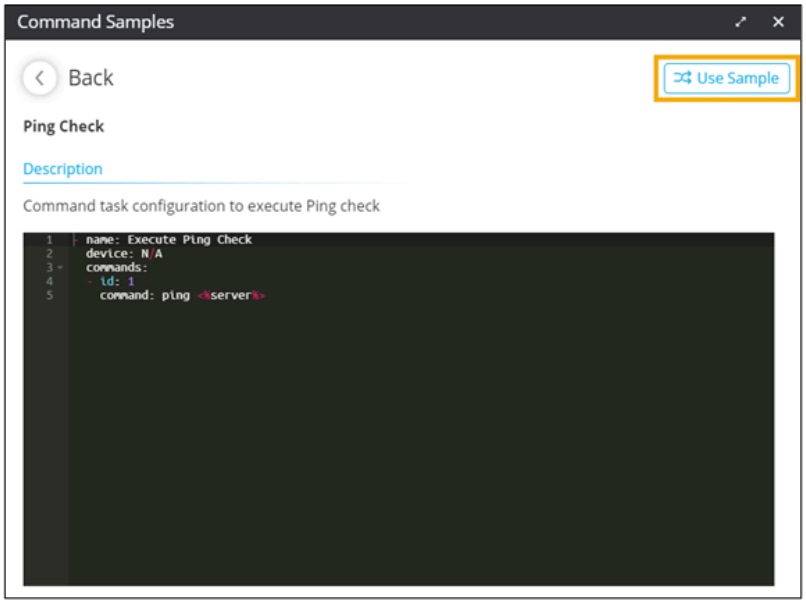


The **Command Samples** window is displayed.

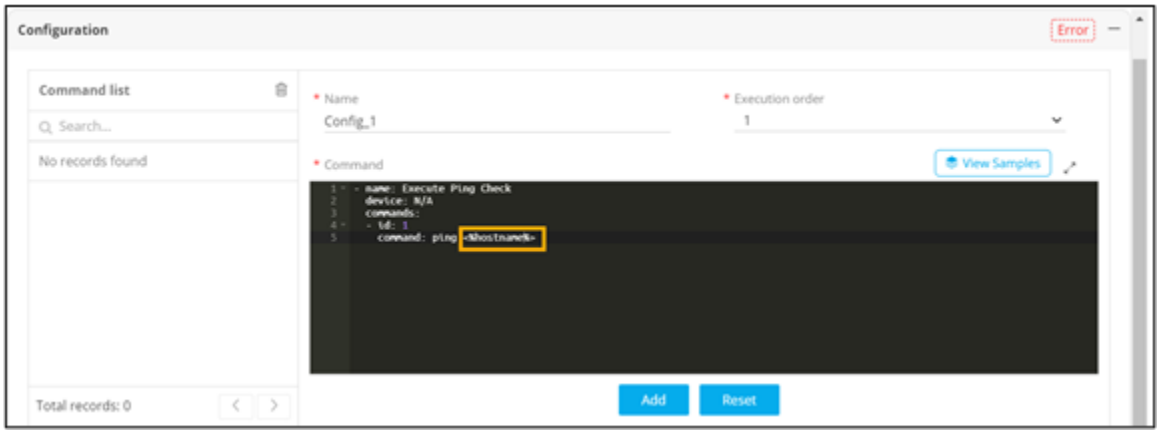


11. Click **View** next to the **Ping Check** command.
12. To use this command, click **Use Sample**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



13. In the **Configuration** section, refer the global variable <%hostname%> defined in the form task.



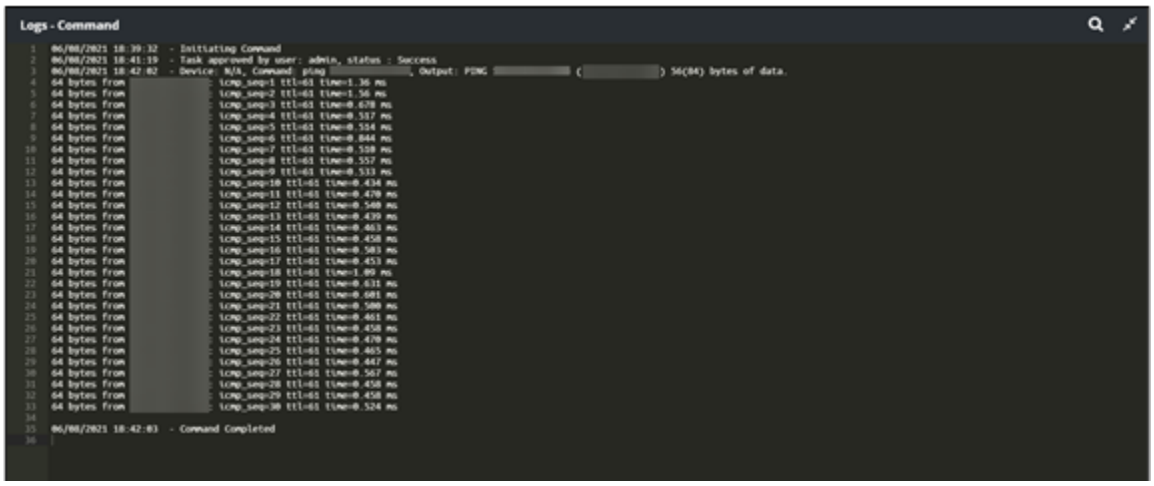
> **Note:** device: N/A implies that the command will be executed on AppViewX server.

14. To add this configuration, click **Add**.
15. Click **Save**.
16. Connect the workflow tasks and enable the workflow.
17. Trigger the workflow.

    The workflow execution page is displayed with the workflow inputs requested at the first stage.
18. Enter the Server IP or Hostname in the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

19. Click **Submit**.
20. In the next stage of workflow execution, to execute the command for performing ping checks on the server, click **Submit**.
    Command is executed with the Logs showing the server response.



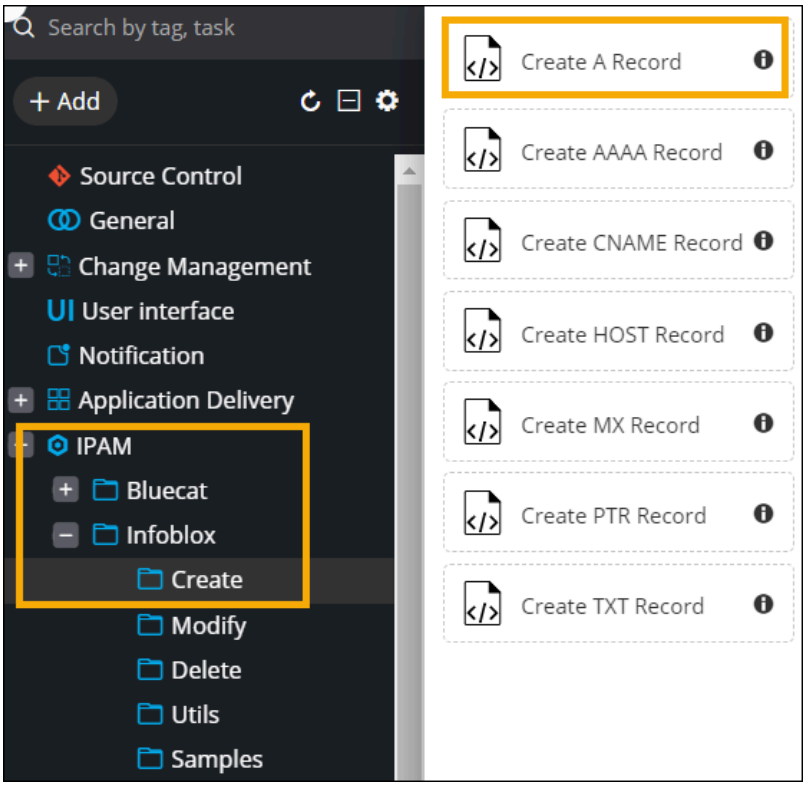
Creating A Record on Infoblox device using prebuilt tasks

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

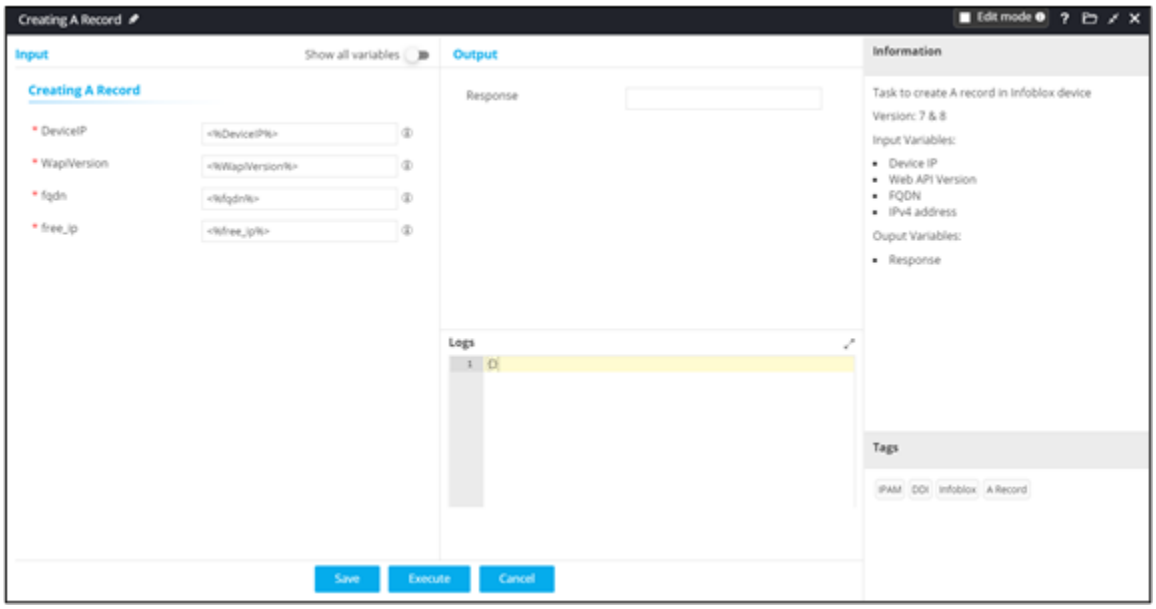
Note: Before creating the workflow, ensure that the device(s) are available in the Device Inventory.

1. Design a new workflow.
2. In the design workspace, from the **IPAM** > **Infoblox** > **Create** folder drag and drop the **Create A Record** prebuilt task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



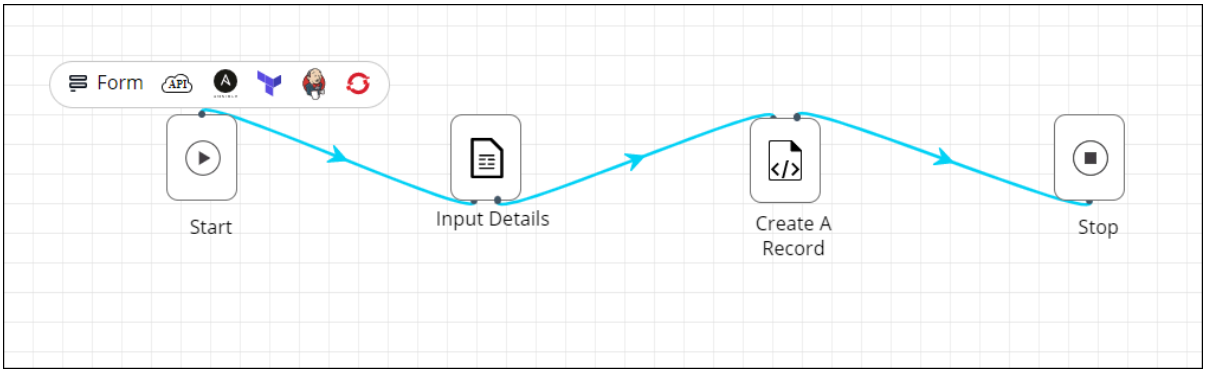
The **Creating A Record** window (Citizen mode) shows the **Input** variables required for this task.



3. To view the API configuration details, in the **Creating A Record** window, select the **Edit mode** checkbox.
4. Click **Save**.
5. Auto generate a self-service form for this workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

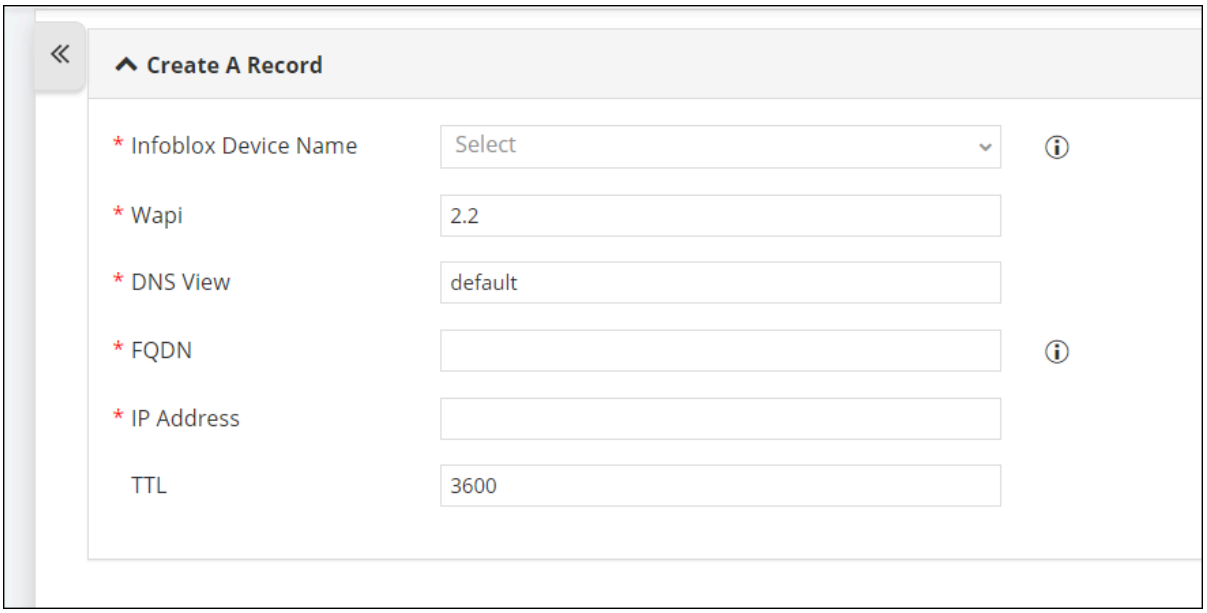
6. Connect the workflow tasks and enable the workflow.



7. Trigger the workflow.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.

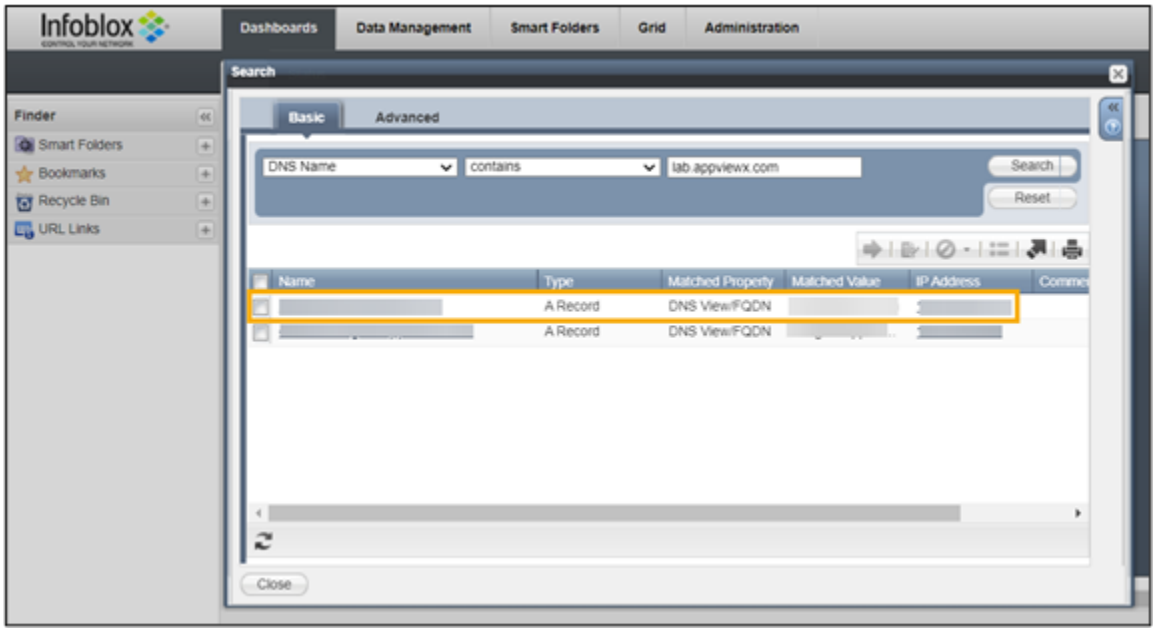
8. Enter the input details in the form.



9. Click **Next**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A record is created on Infoblox.



Creating a VIP with Incident Ticket

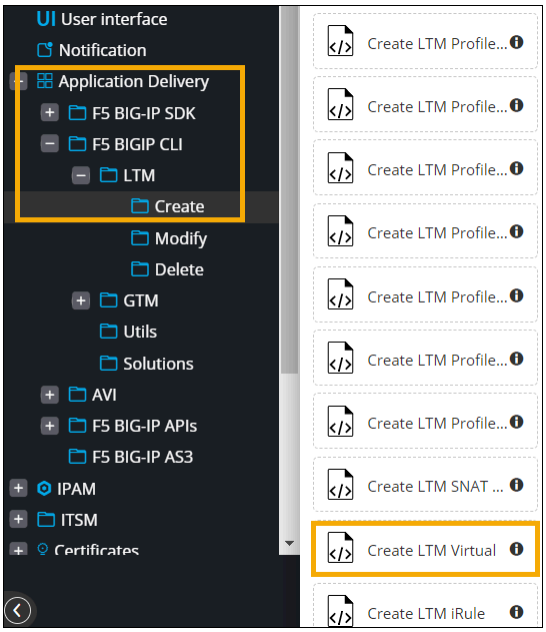
The ITSM folder allows you to leverage the prebuilt tasks for designing ITSM automation workflows.

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

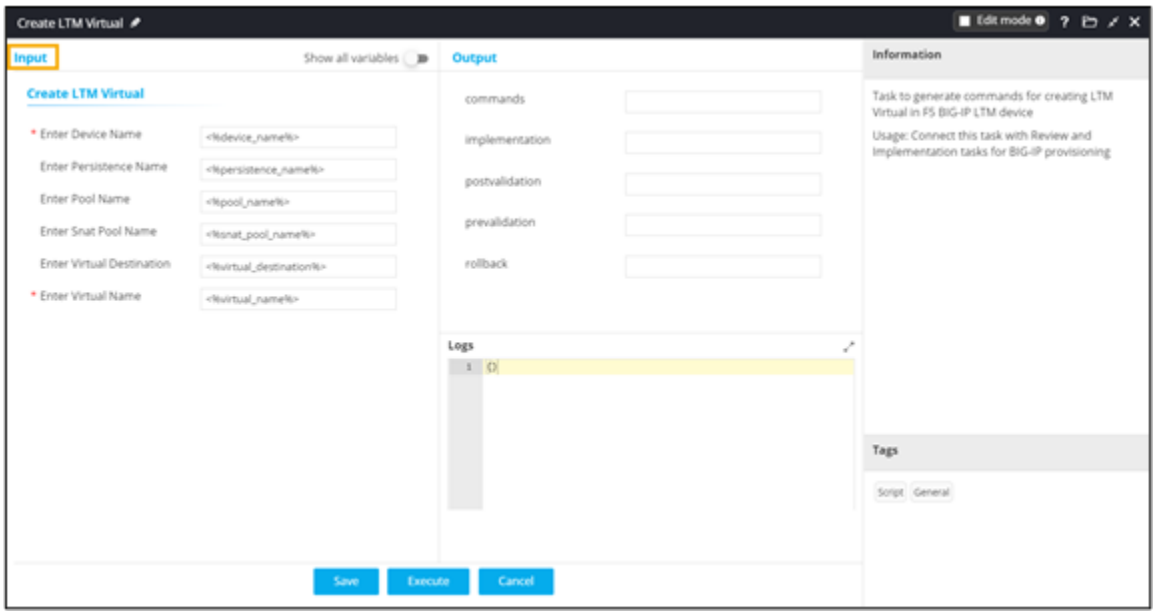
To create a VIP with Incident ticket:

1. Design a new workflow.
2. To create a virtual LTM server, from the **Application Delivery** > **F5 BIG-IP CLI Commands** > **LTM** > **Virtual** folder, drag and drop the prebuilt **Create LTM Virtual task**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

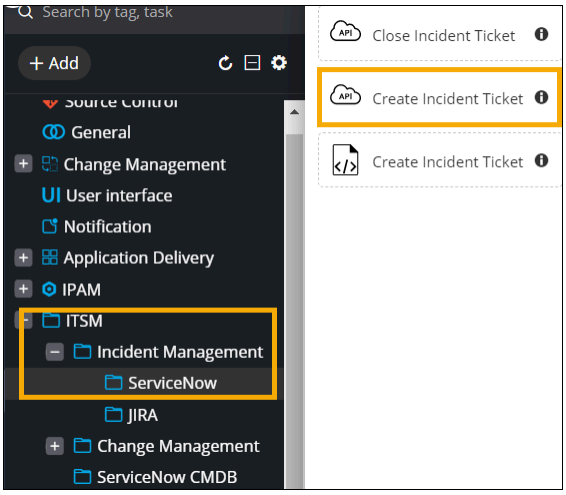


The **Create LTM Virtual** window (Citizen mode) shows the **Input** variables required for this task.

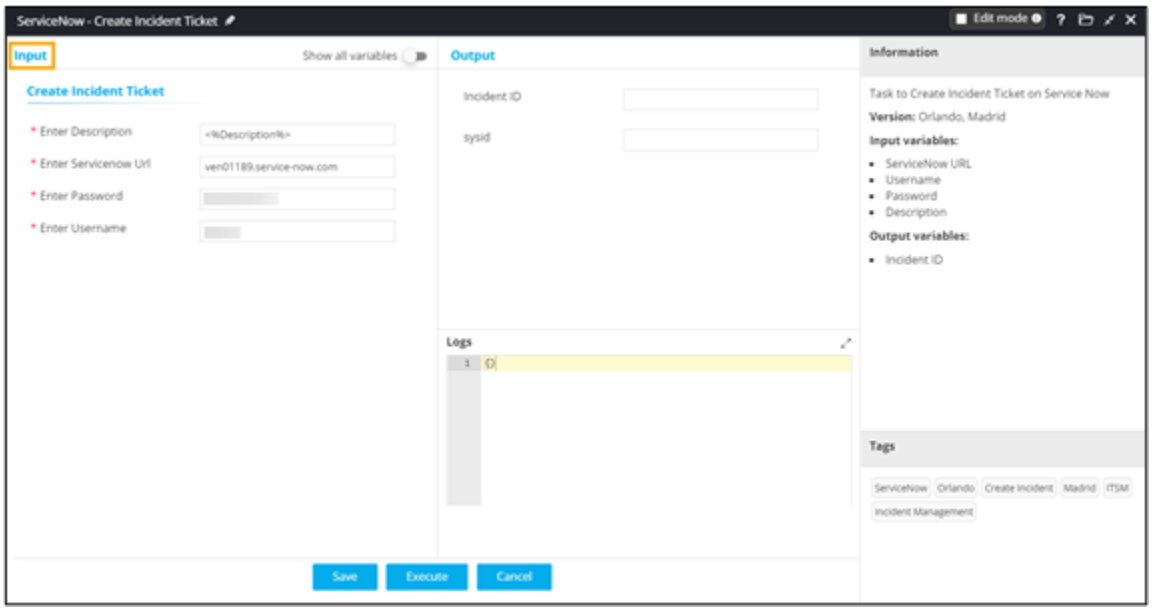


3. To view or edit the task configurations, in the **Create LTM Virtual** window, from the top right corner, select the **Edit mode** checkbox.
4. Click **Save**.
5. From the **ITSM** > **Incident Management** > **ServiceNow** folder, drag and drop the **Create Incident Ticket** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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



7. Click **Save**.
8. From the **User Interface** section in the left menu, drag and drop the **Review** task.
9. Click **Save**.
10. From the **General** section in the left menu, drag and drop the **Prevalidation** task.
11. Click **Save**.
12. From the **General** section in the left menu, drag and drop the **Implementation** task.
13. Click **Save**.
14. From the **General** section in the left menu, drag and drop the **Postvalidation** task.
15. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

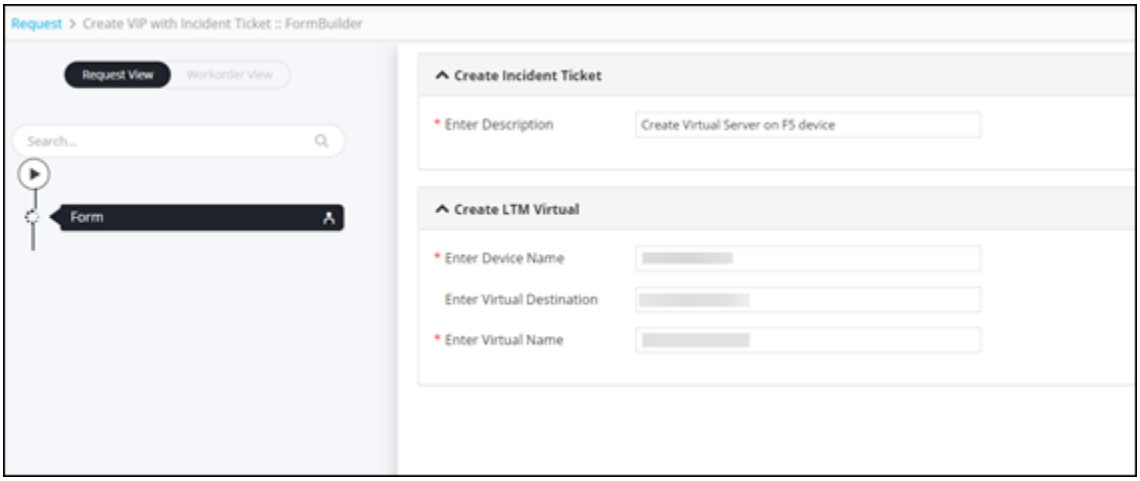
16. Auto generate self-service form for this workflow.
17. Connect the workflow tasks.
18. To add a rollback workflow, from the menu on the top, click **Rollback**.

19. From the **General** section in the left menu, drag and drop the **Rollback** task.
20. Click **Save**.
21. Connect the rollback workflow tasks and enable the workflow.

> **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 .

22. Trigger the workflow.

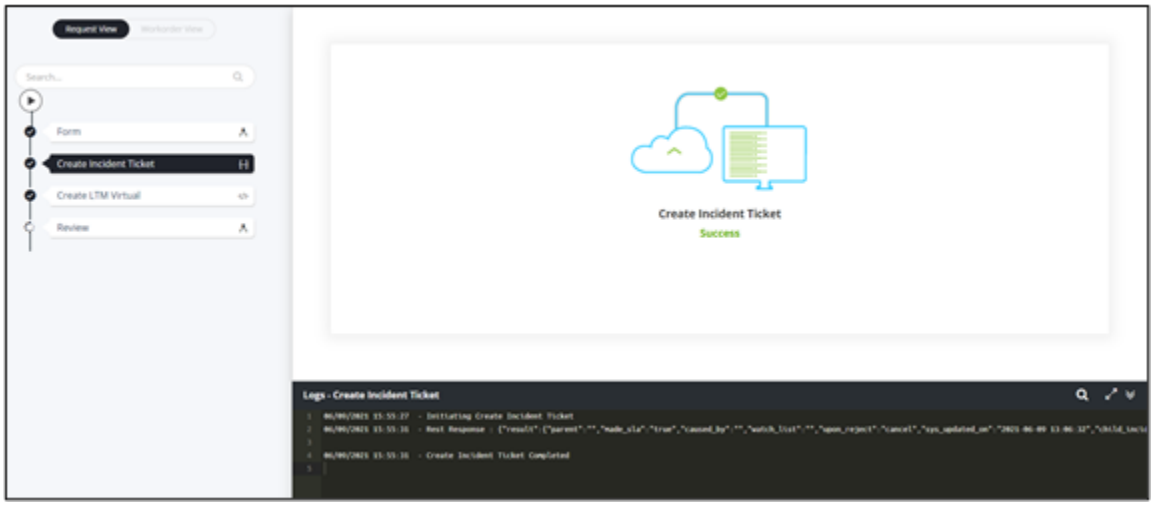
    The workflow execution page is displayed with the workflow inputs requested at the first stage.
23. Enter the input details in the form.

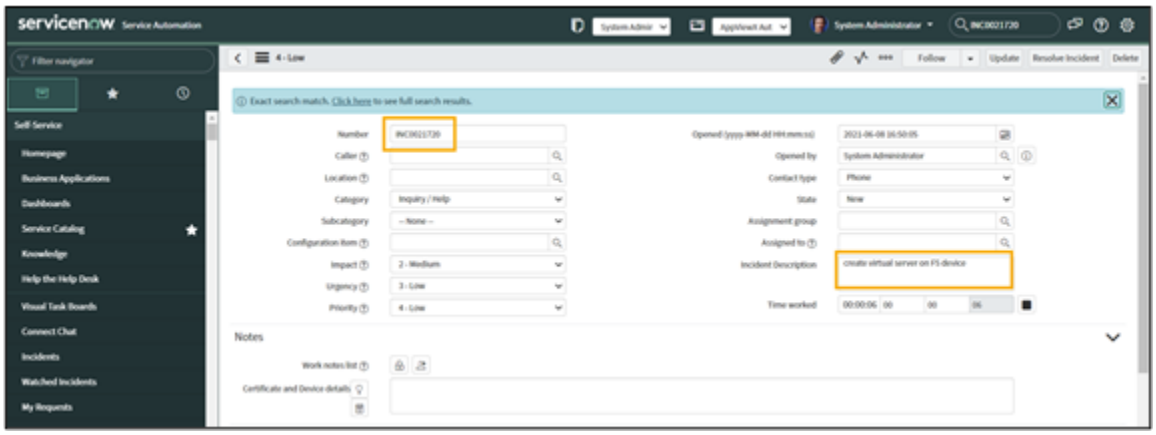


24. Click **Next**.

    Incident Ticket is created on ServiceNow.

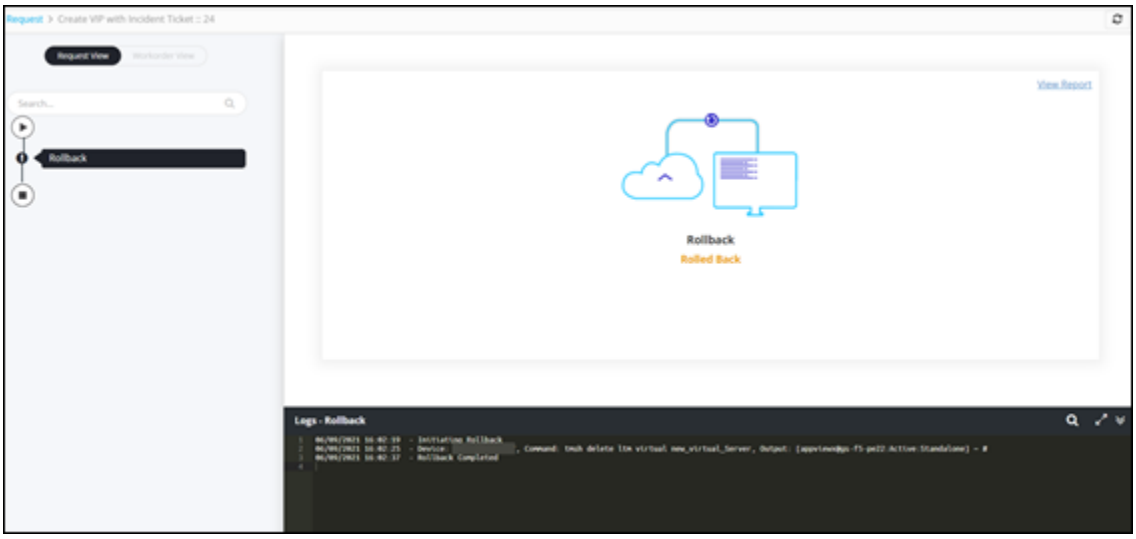
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



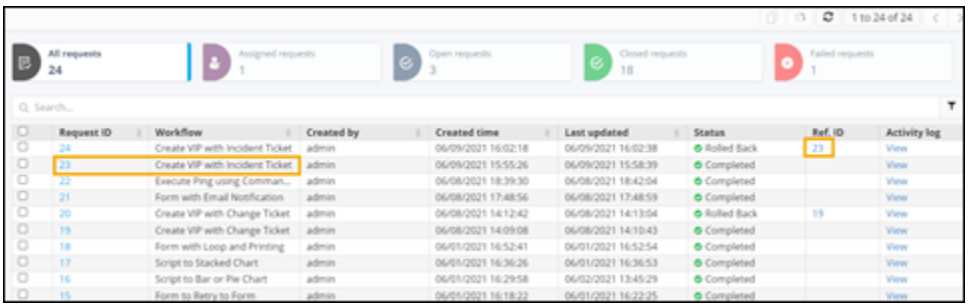


25. At the **Review** stage, to execute the implementation, prevalidation, rollback, and postvalidation commands, click **Submit**.
26. To execute the rollback workflow, on the Request :: All page, select the checkbox next to the workflow **Request ID**.
27. From the upper right corner of the screen, click **(Rollback)** icon.
    Rollback request to delete the virtual server that was created, is executed successfully.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



28. To view the request ID of the rollback workflow, navigate to the Request :: All page.
    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

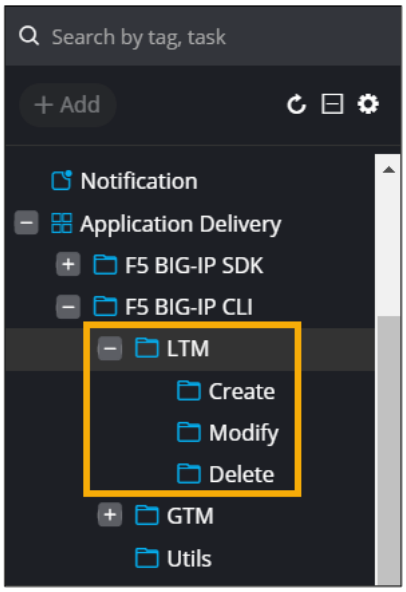
This section describes some of the workflows/tasks for F5 BIG-IP provisioning available in the Workflow Studio.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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

LTM

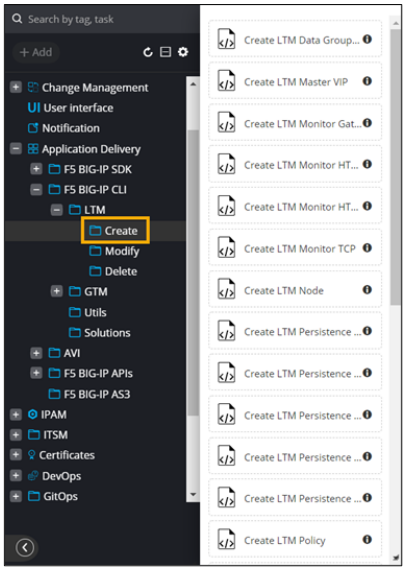
The workflows and tasks to create, modify, and delete LTM objects and devices on F5 BIG-IP are available under **Application Delivery** > **F5 BIG-IP CLI** > **LTM**.



LTM - Create

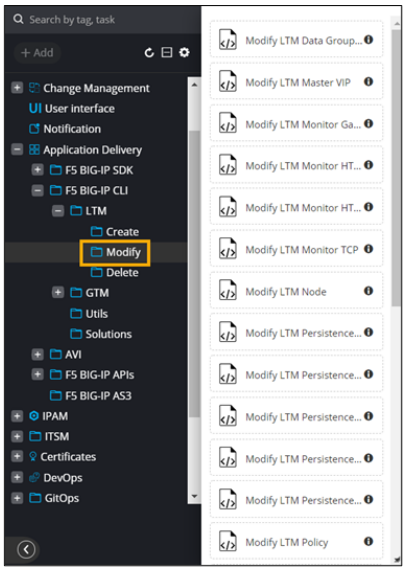
The **Create** folder under **LTM** contains prebuilt tasks that can be leveraged to create LTM objects on F5 BIG-IP.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



LTM - Modify

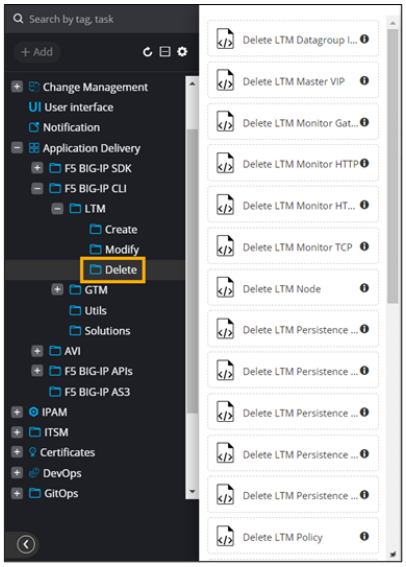
The **Modify** folder under **LTM** contains prebuilt tasks that can be leveraged to modify LTM objects on F5 BIG-IP.



LTM - Delete

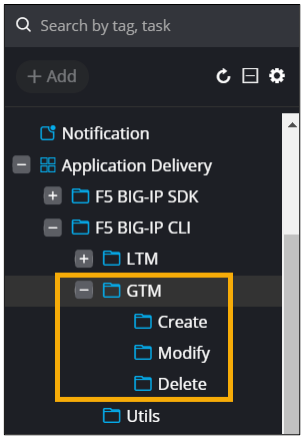
The **Delete** folder under **LTM** contains prebuilt tasks that can be leveraged to delete LTM objects on F5 BIG-IP.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



GTM

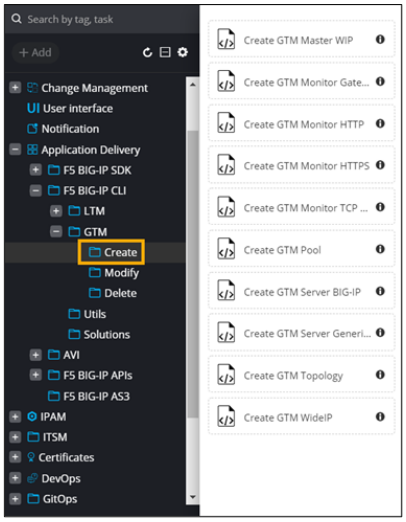
The workflows and tasks to create, modify, and delete GTM objects and devices on F5 BIG-IP are available under **Application Delivery** > **F5 BIG-IP CLI** > **GTM**.



GTM - Create

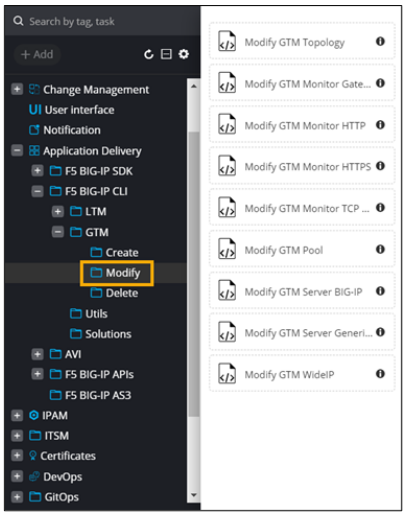
The **Create** folder under **GTM** contains prebuilt tasks that can be leveraged to create GTM objects on F5 BIG-IP.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



GTM - Modify

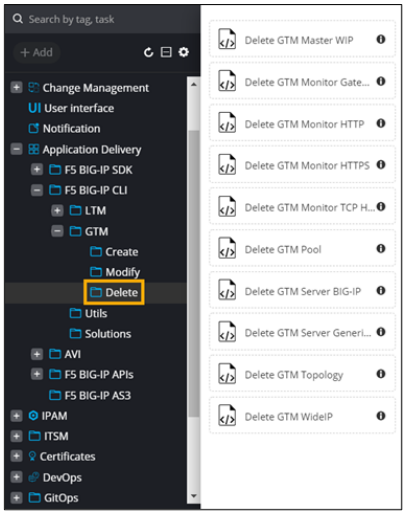
The **Modify** folder under **GTM** contains prebuilt tasks that can be leveraged to modify GTM objects on F5 BIG-IP.



GTM - Delete

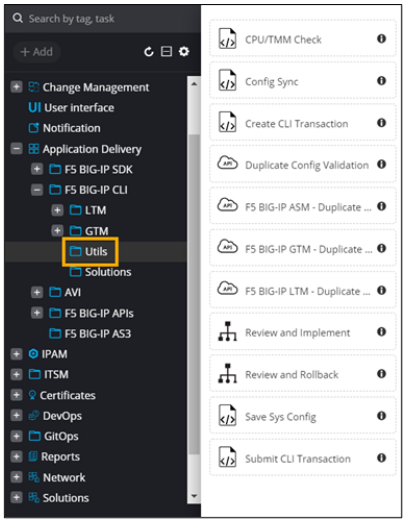
The **Delete** folder under **GTM** contains prebuilt tasks that can be leveraged to delete GTM objects on F5 BIG-IP.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Utils

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

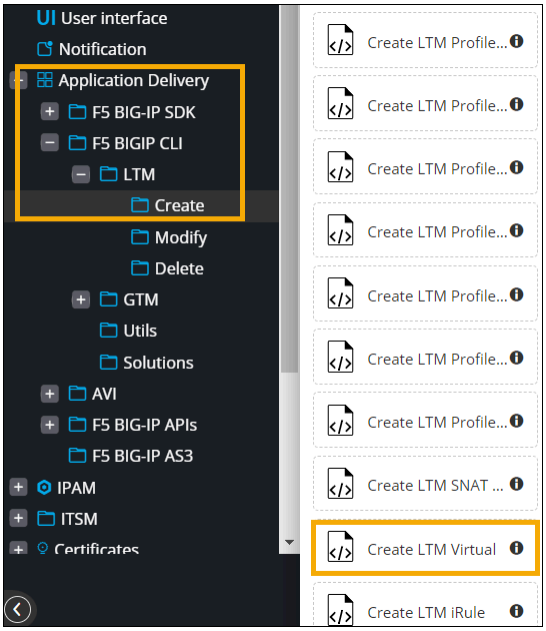


Creating a LTM Virtual Server on F5 BIG-IP

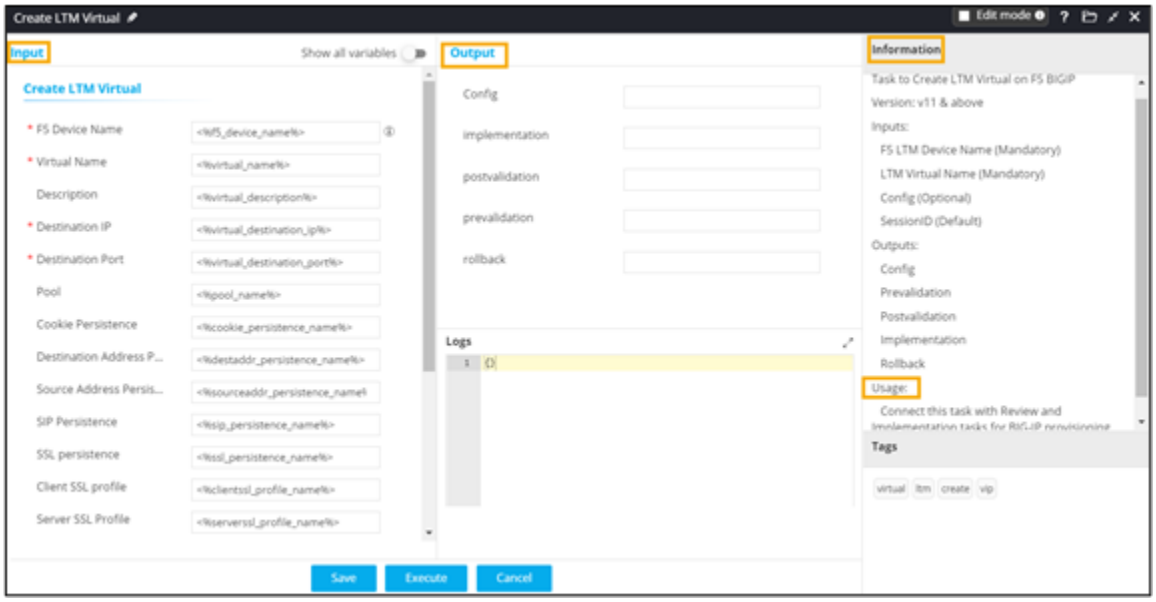
You can leverage the prebuilt LTM tasks to design a workflow for creating a LTM Virtual Server on F5 BIG-IP, add a CPU/TMM check, and also design a rollback workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Design a new workflow.
2. In the design workspace, from the **Application Delivery** > **F5 BIG-IP CLI** > **LTM** > **Create** folder, drag and drop the **Create LTM Virtual** task.



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.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

5. Auto generate a self-service form for this workflow.
6. Connect the workflow tasks and enable the workflow.
7. Trigger the workflow.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.
8. Enter the details in the input form.

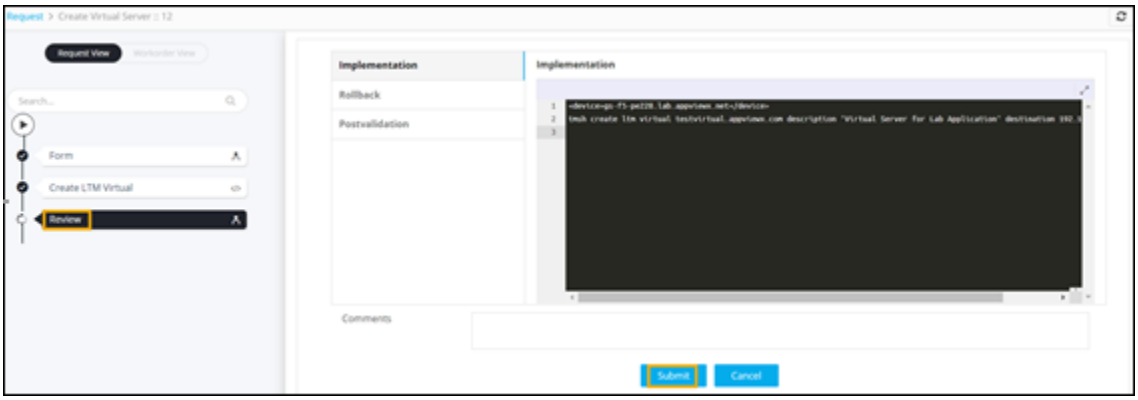
> **Note:** Ensure that the device is available in the Device Inventory.

9. Click **Next**.
10. In the **Confirmation** pop-up window, click **Ok**.

    The **Create LTM Virtual** task is successfully completed.

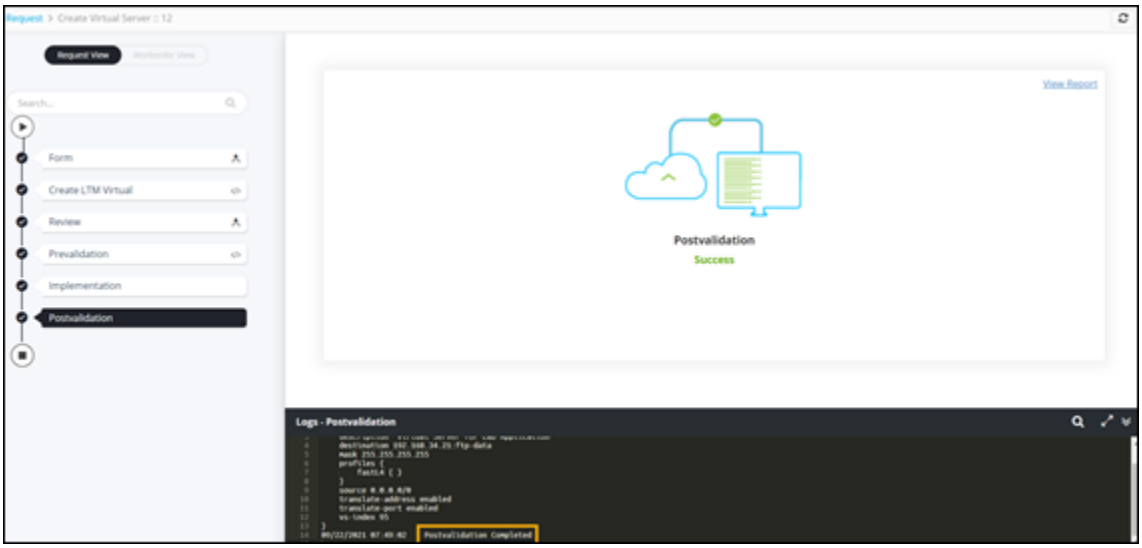
11. At the **Review** stage, click **Submit**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



12. In the **Confirmation** pop-up window, click **Ok**.

    Workflow is successfully executed.

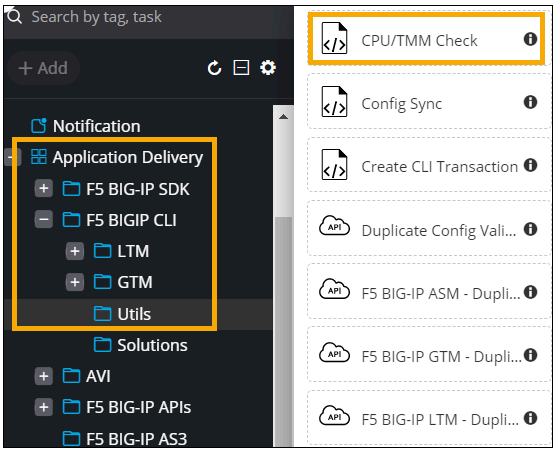


- Performing a CPU/TMM Check
- Adding a Rollback Workflow

Performing a CPU/TMM Check

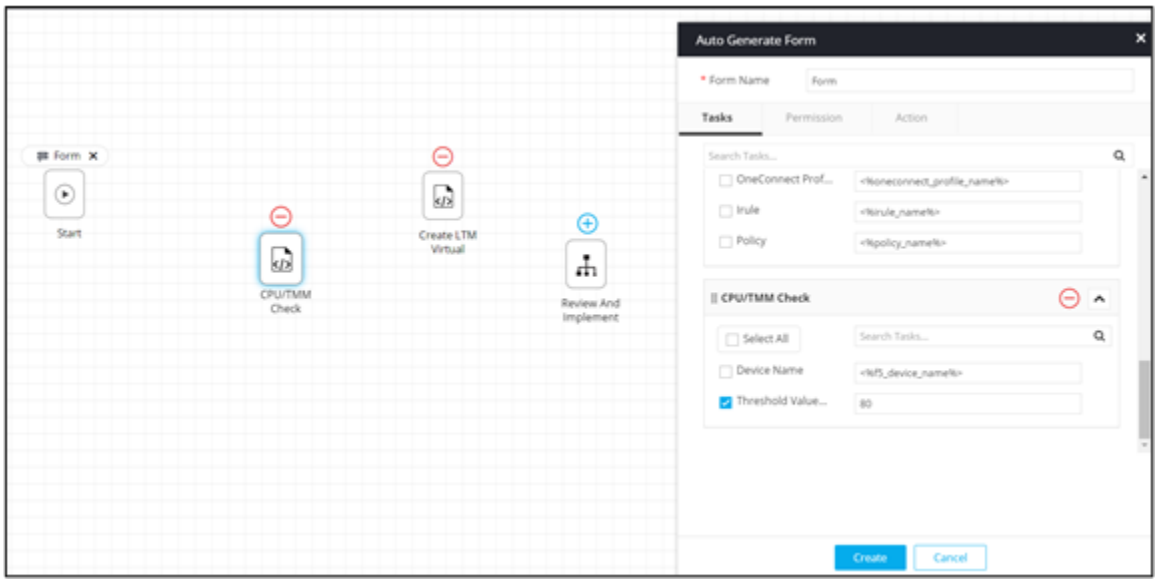
1. Design a workflow to create a LTM virtual server on F5 BIG-IP.
2. In the design workspace, from the **Utils** folder, under **F5 BIG-IP CLI**, drag and drop the **CPU/TMM Check** task. This task performs a CPU/TMM check on F5 BIG-IP.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

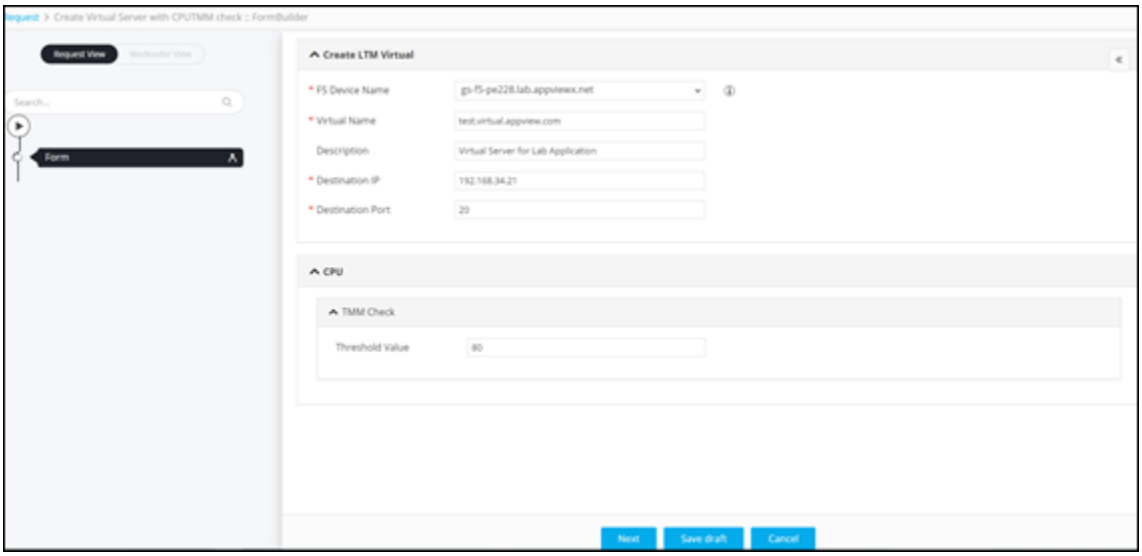
3. Click **Save**.
4. Auto-generate a self-service form for this workflow.



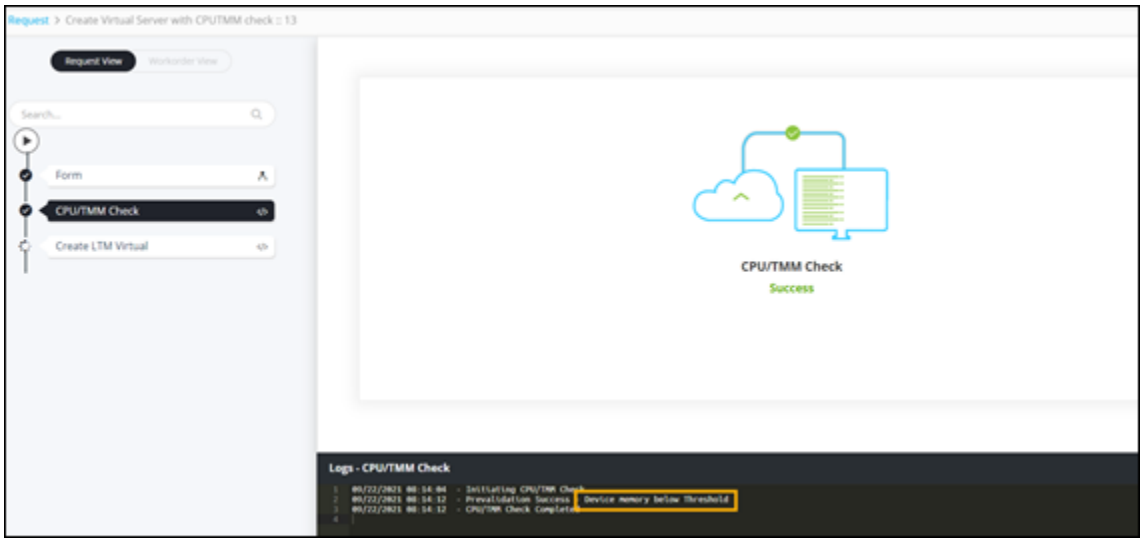
5. Connect the workflow tasks and enable the workflow.
6. Trigger the workflow.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.
7. Enter the details in the input form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

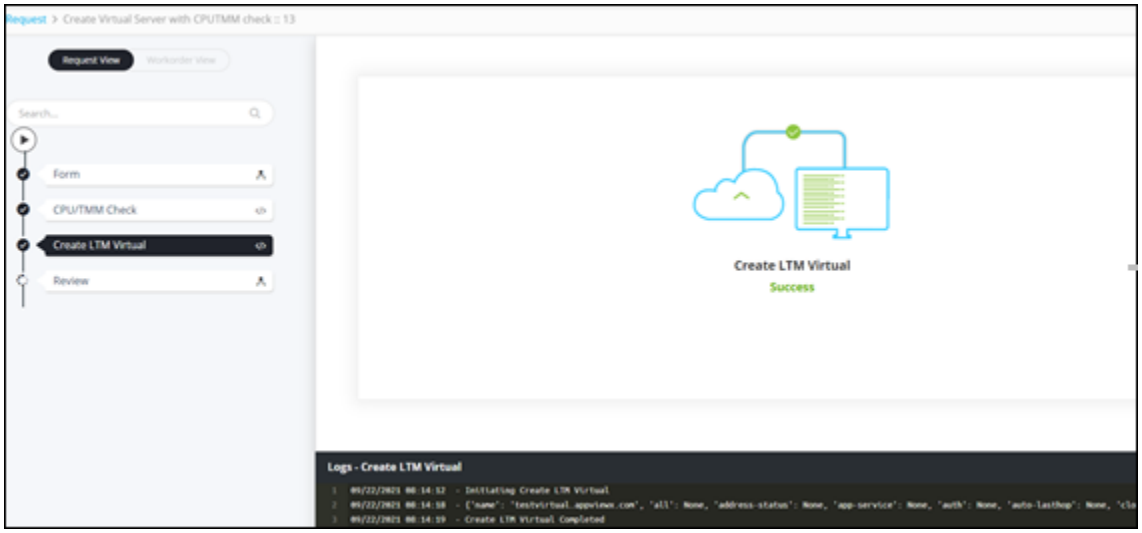


8. Click **Next**.
   - **CPU/TMM Check** is successful - Device memory is below threshold.



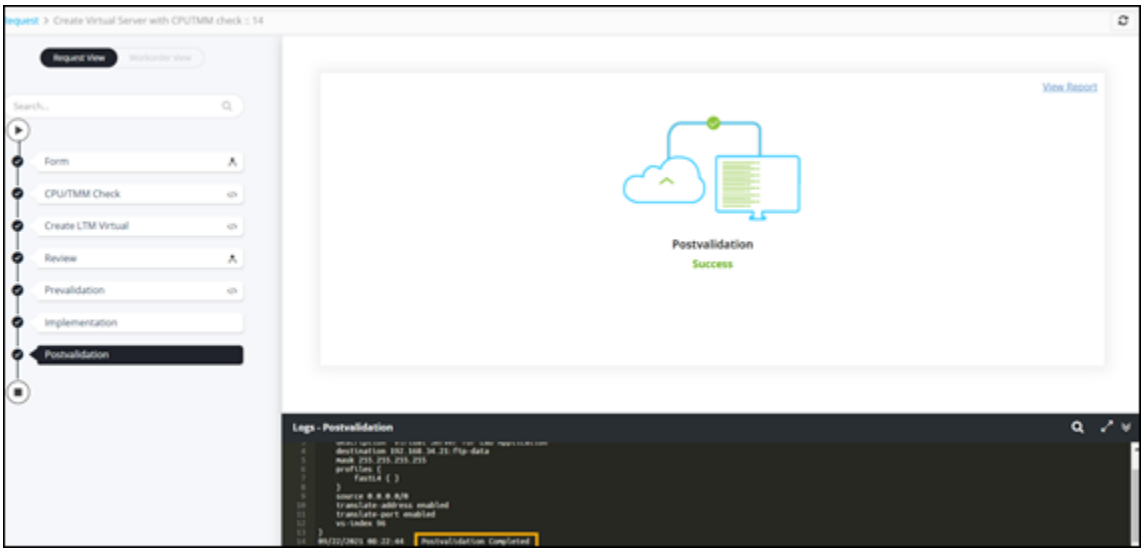
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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



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

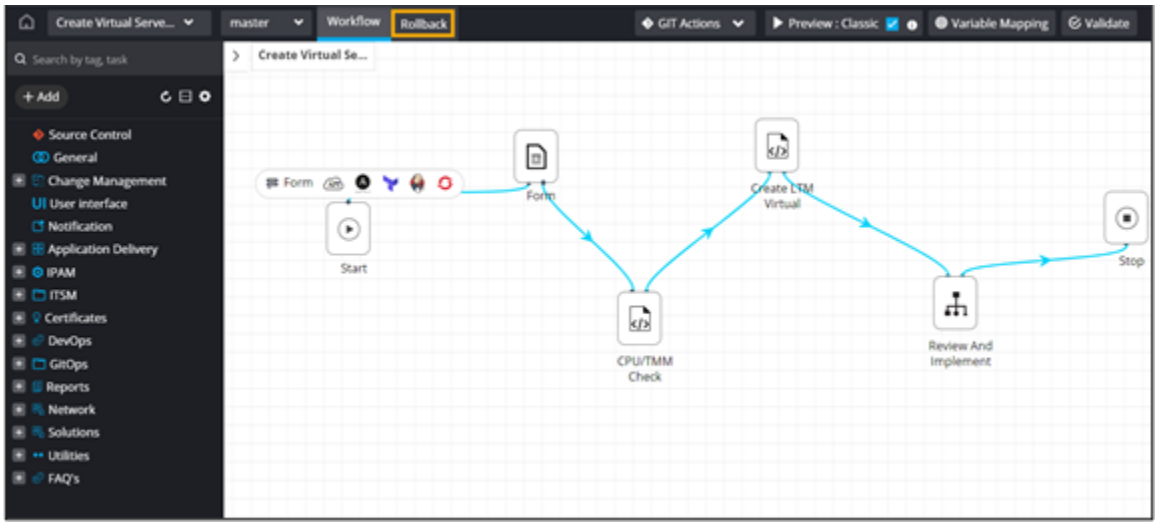
   The workflow is executed successfully.



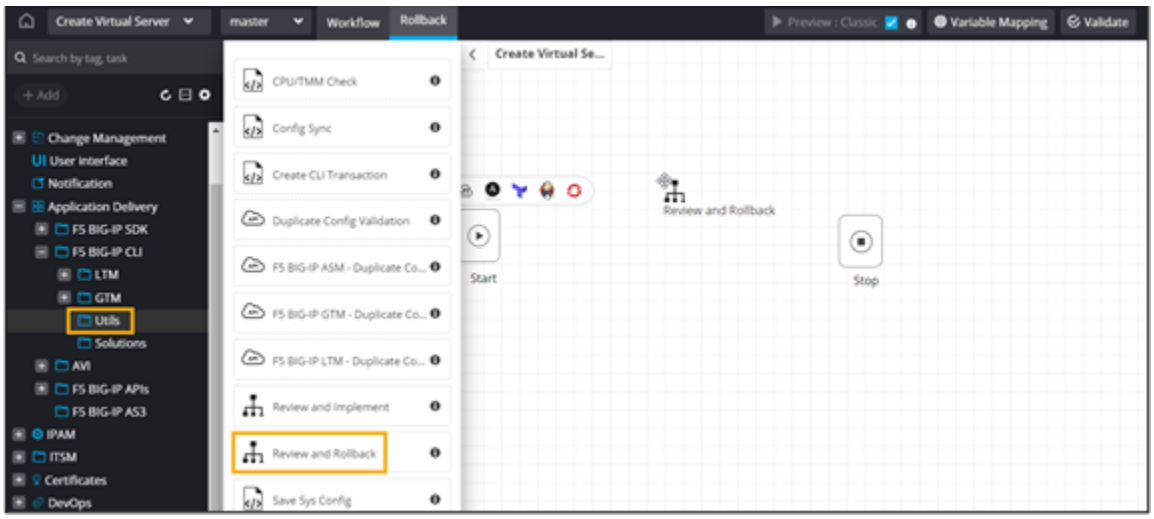
Adding a Rollback Workflow

1. Design a workflow to create a LTM virtual server on F5 BIG-IP with CPU/TMM check.
2. In the design workspace, from the menu on the top, click **Rollback**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



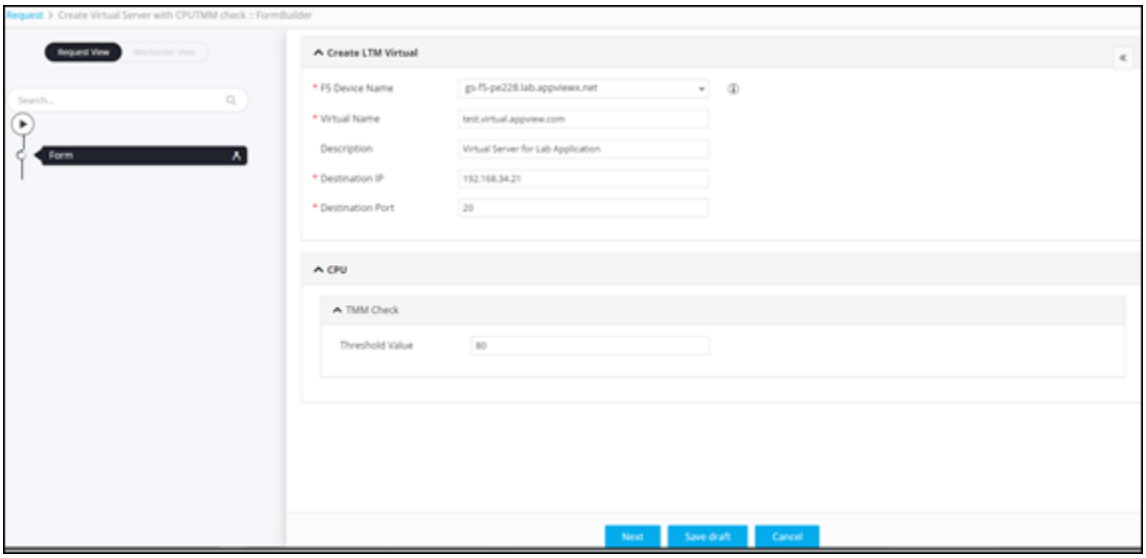
3. From the **Utils** folder in the left menu, drag and drop the **Review and Rollback** workflow.



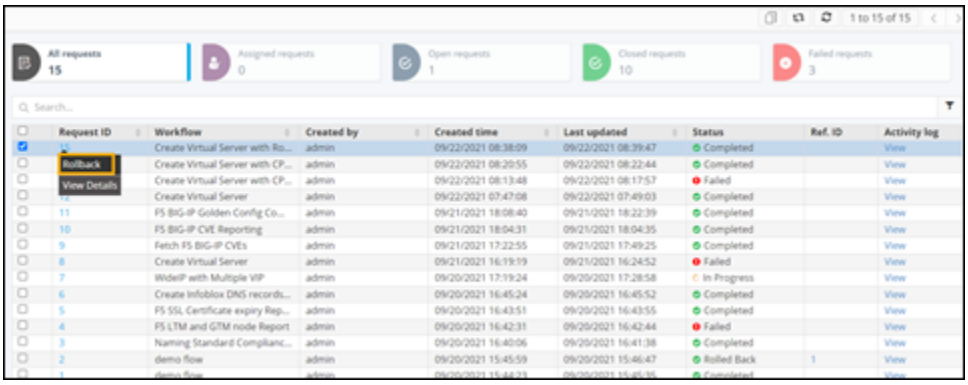
4. Connect the workflow tasks and enable the workflow.
5. Trigger the workflow.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.
6. Enter the details in the input form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



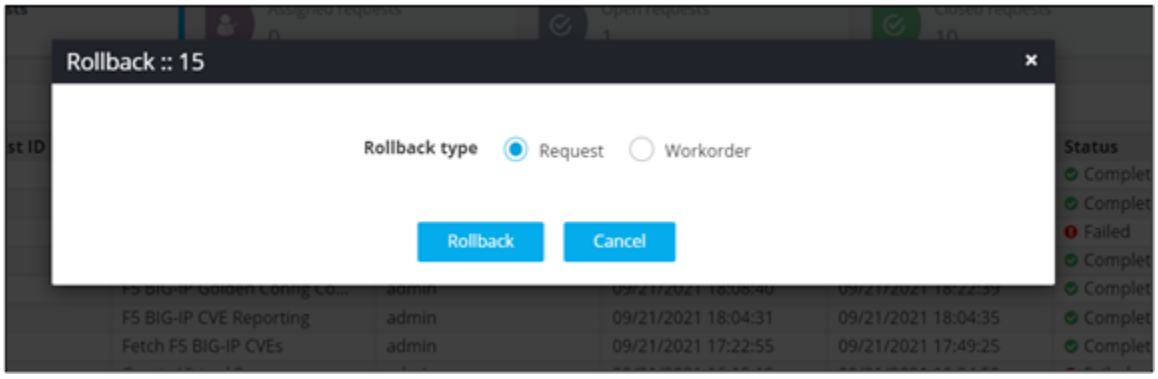
7. At the **Review** stage, click **Submit**.
   The workflow is executed successfully.
8. To trigger the rollback workflow, on the **Request :: All** page, right-click on the **Request ID** of the workflow.
9. From the options displayed, select **Rollback**.



> **Tip:** You can also select the workflow and click **(Rollback)** icon from the command bar in the top right corner of the screen.

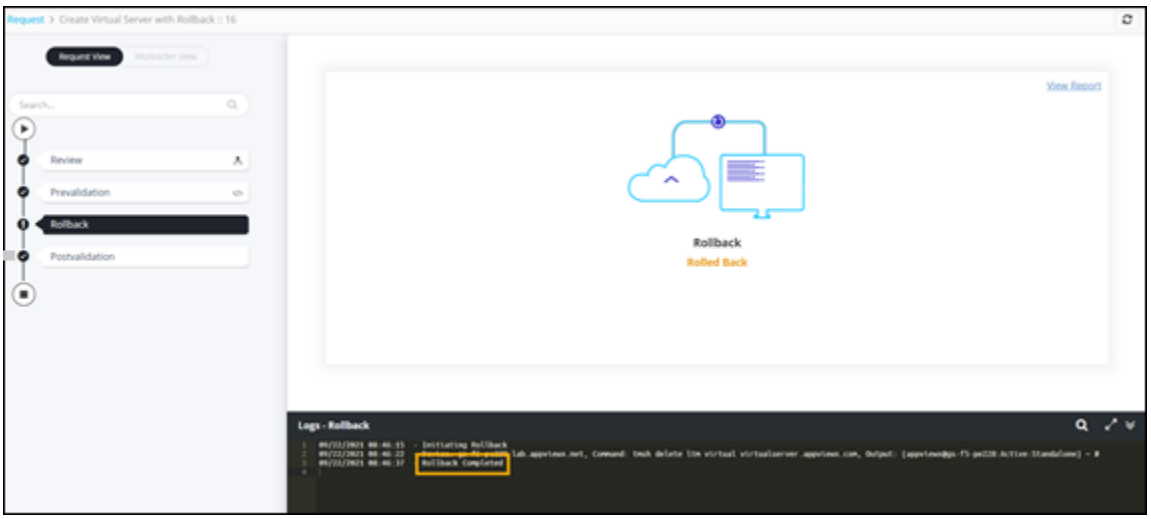
10. In the **Rollback** pop-up window, click **Yes** and select **Rollback type** as **Request**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



11. Click **Rollback**.
12. At the **Review** stage, click **Submit**.

    Rollback completed.



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

AVI Solutions

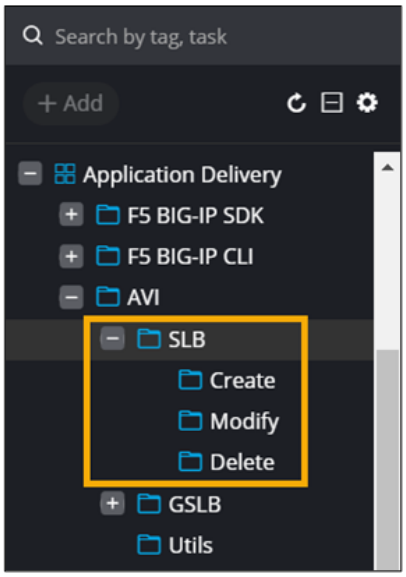
This section describes some of the workflows/tasks for AVI provisioning available in the Workflow Studio.

- SLB
- GSLB
- Utils

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

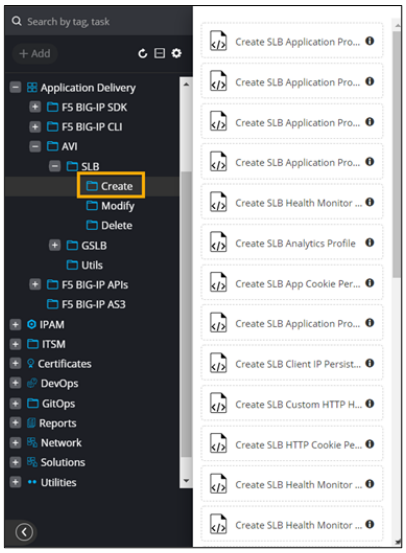
SLB

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



SLB - Create

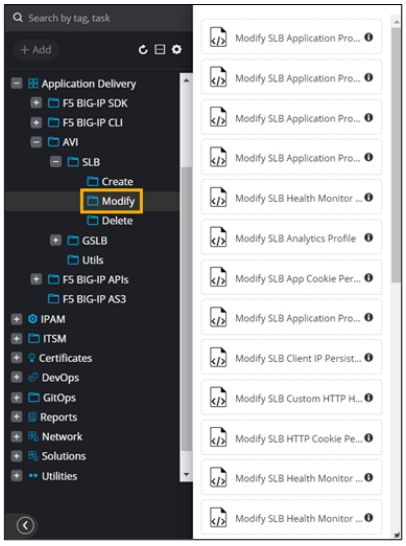
The **Create** folder under **SLB** contains prebuilt tasks that can be leveraged to create SLB objects on AVI.



SLB - Modify

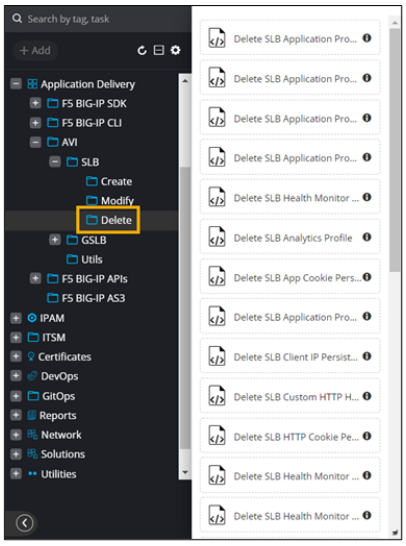
The **Modify** folder under **SLB** contains prebuilt tasks that can be leveraged to modify SLB objects on AVI.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



SLB - Delete

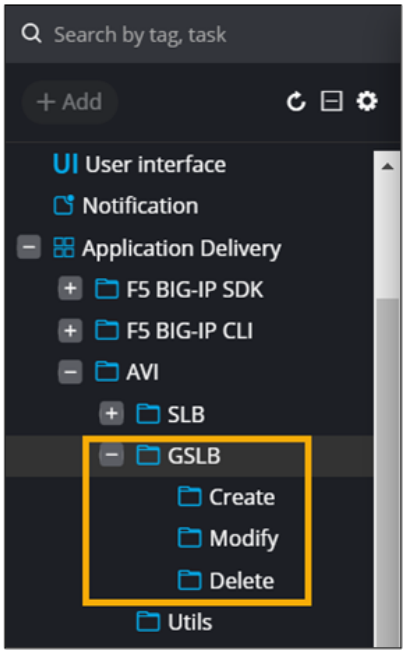
The **Delete** folder under **SLB** contains prebuilt tasks that can be leveraged to delete SLB objects on AVI.



GSLB

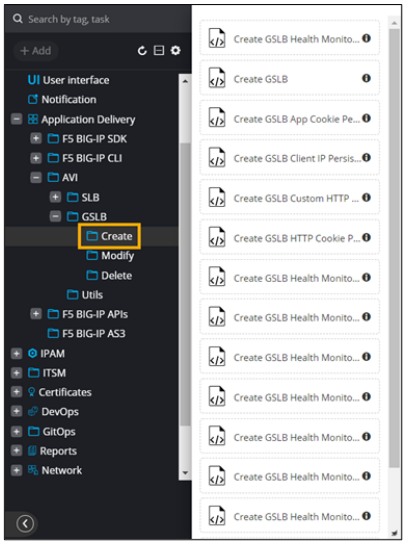
The workflows and tasks to create, modify, and delete GSLB objects and devices on AVI are available under **Application Delivery** > **AVI** >**GSLB** GSLB.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



GSLB - Create

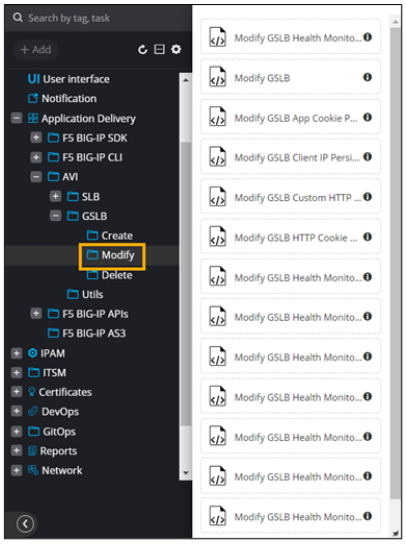
The **Create** folder under **GSLB** contains prebuilt tasks that can be leveraged to create GSLB objects on AVI.



GSLB - Modify

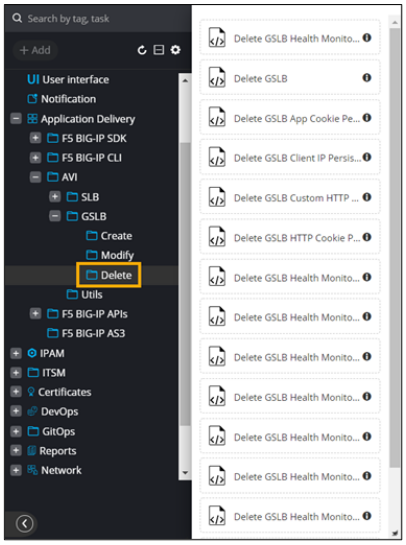
The **Modify** folder under **GSLB** contains prebuilt tasks that can be leveraged to modify GSLB objects on AVI.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



GSLB - Delete

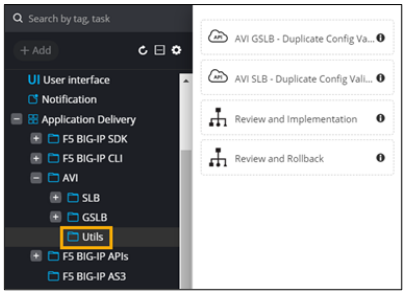
The **Delete** folder under **GSLB** contains prebuilt tasks that can be leveraged to delete GSLB objects on AVI.



Utils

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

Copyright © 2025 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.

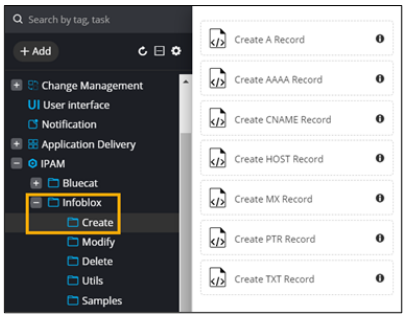
- Infoblox Solutions
- Bluecat Solutions

Infoblox Solutions

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

Create

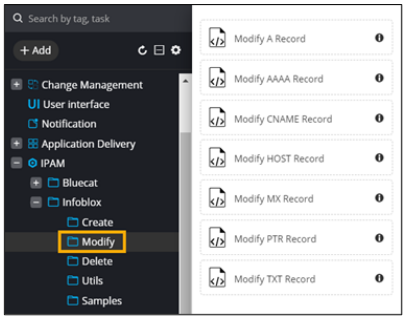
The **Create** folder under **Infoblox** contains prebuilt tasks that can be leveraged to create DNS records such as A Record, CNAME record, and PTR Record on Infoblox.



Modify

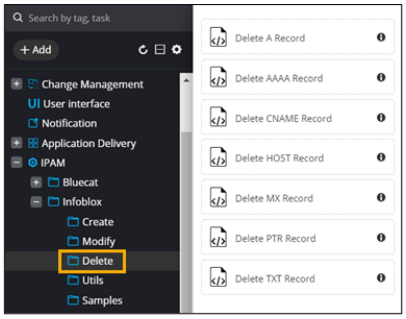
The **Modify** folder under **Infoblox** contains prebuilt tasks that can be leveraged to modify DNS records such as A Record, CNAME record, and PTR Record on Infoblox.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



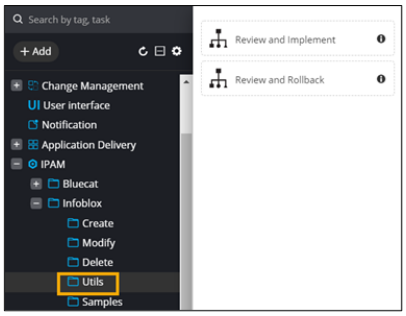
Delete

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.



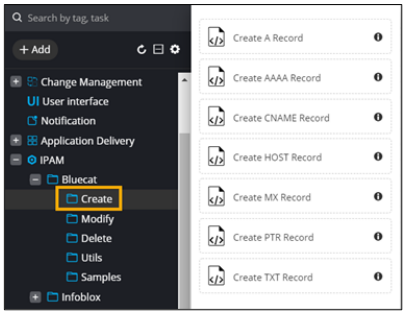
Bluecat Solutions

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

Create

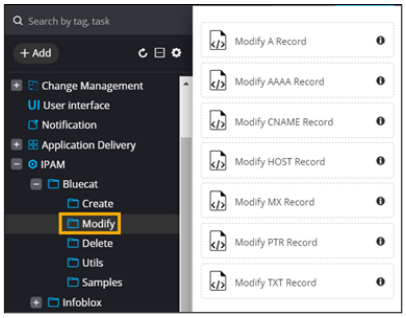
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.



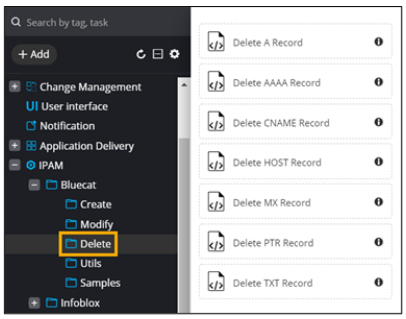
Modify

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

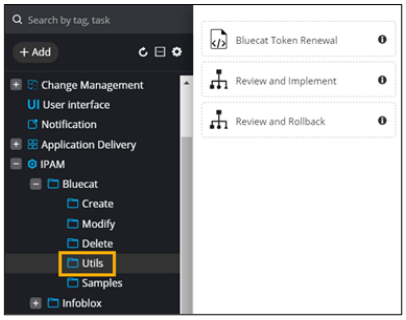
The **Delete** folder under **Bluecat** contains prebuilt tasks that can be leveraged to delete DNS records such as A Record, CNAME record, and PTR Record on Bluecat.



Utils

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

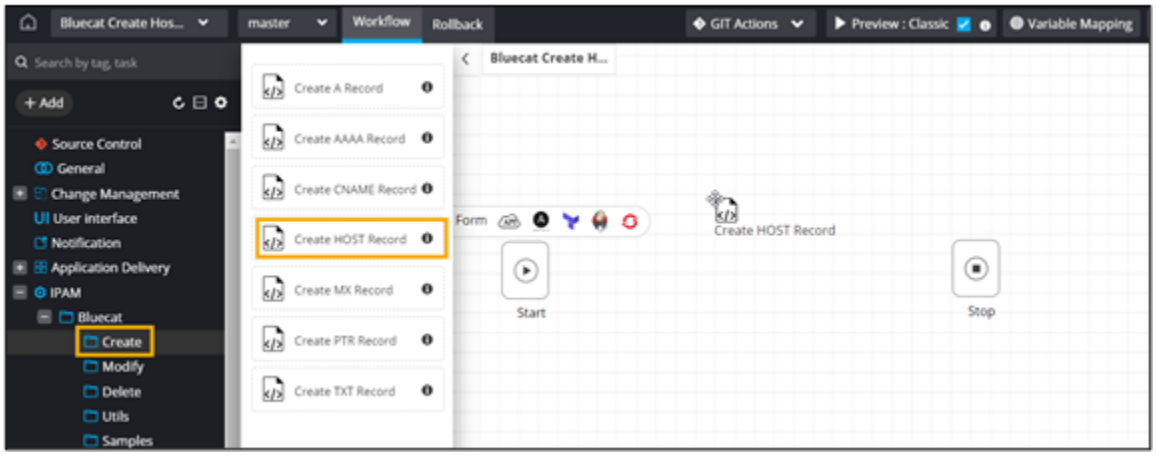
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



• Creating a HOST Record in Bluecat

Creating a HOST Record in Bluecat

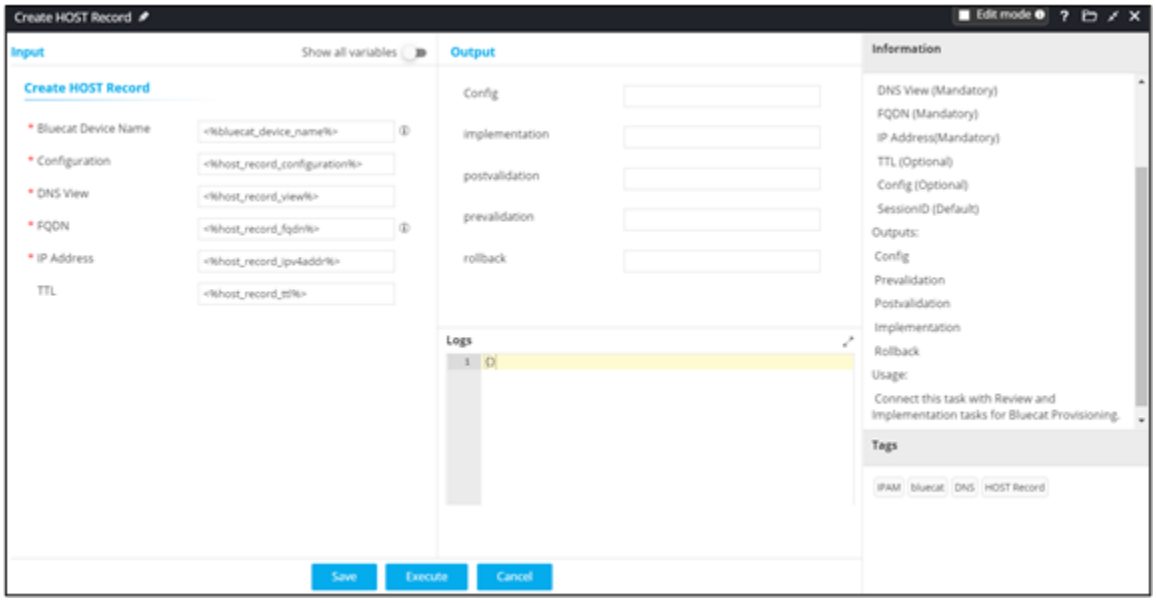
1. Design a new workflow.
2. In the design workspace, from **IPAM** > **Bluecat** > **Create** folder, drag and drop the **Create HOST Record** task.



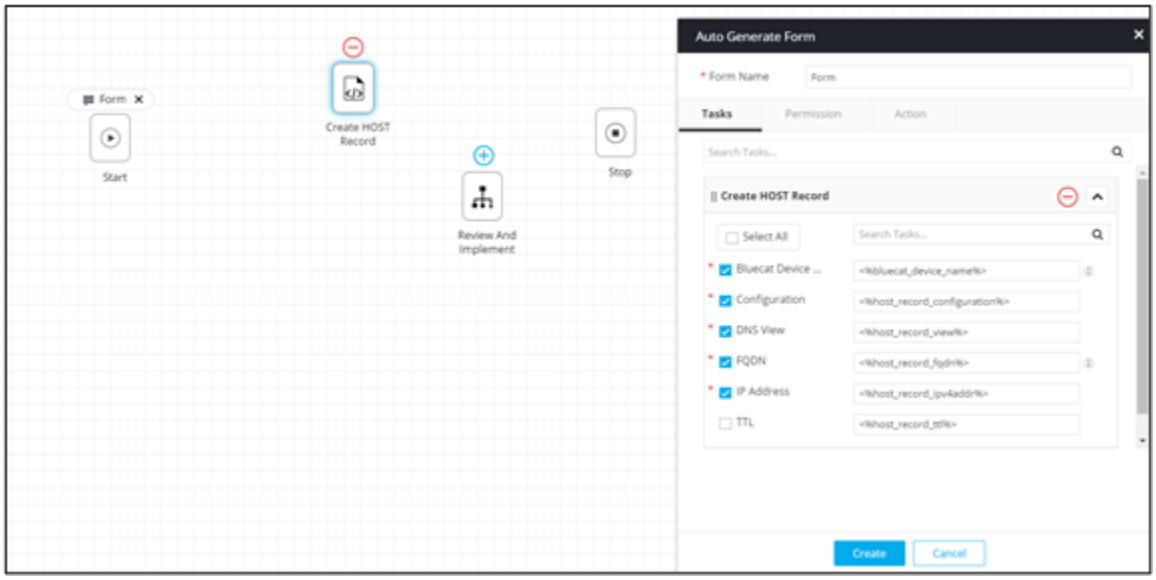
.

Note: 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

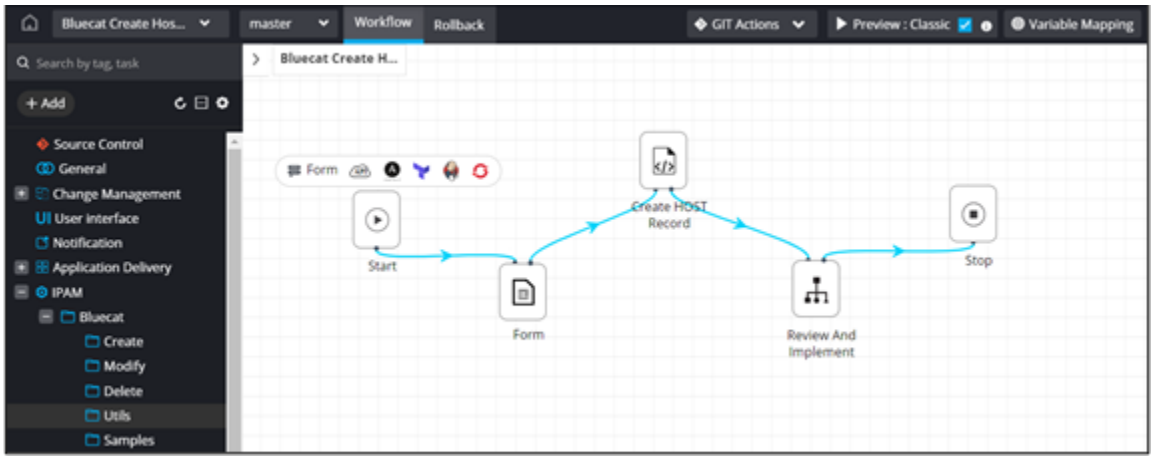


3. Click **Save**.
4. From the **Utils** folder in the left menu, under **Bluecat**, drag and drop the **Review and Implement** workflow.
5. Auto-generate a self-service form for this workflow.



6. Connect the workflow tasks and enable the workflow.

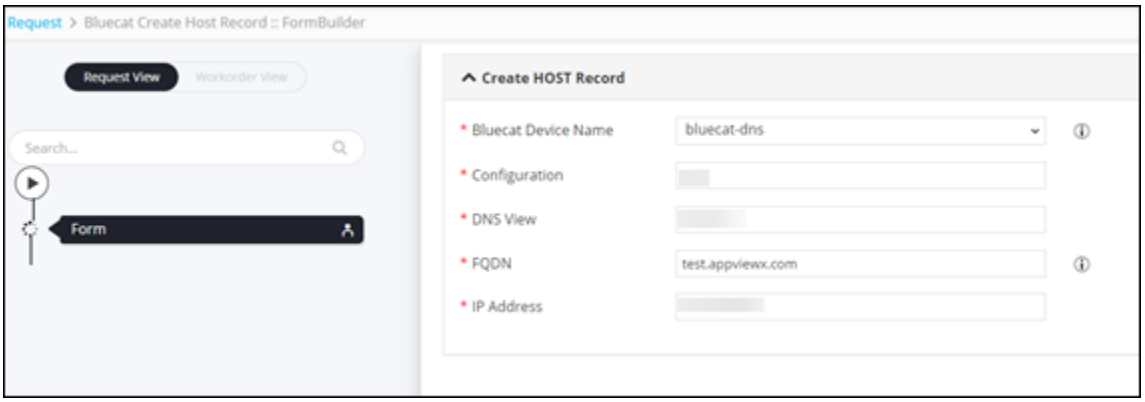
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



7. Trigger the workflow.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.

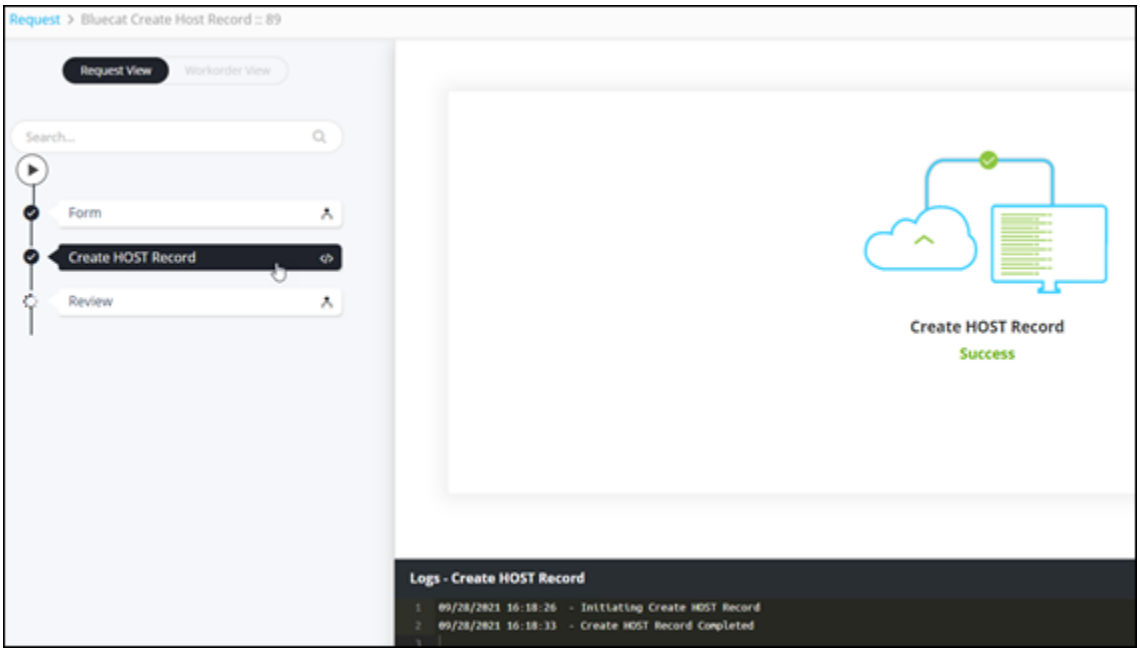
8. Enter the details in the input form.



9. Click **Next**.

   The **HOST Record** is created.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



10. At the **Review** stage, click **Submit**.
    The workflow is executed successfully.

Change Automation

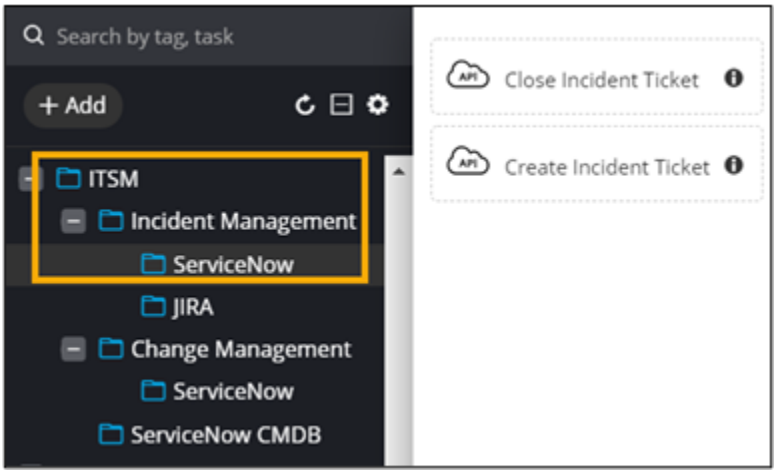
This folder consists of prebuilt tasks and workflows that allow you to integrate with ITSM tools. AppViewX supports the following vendors:

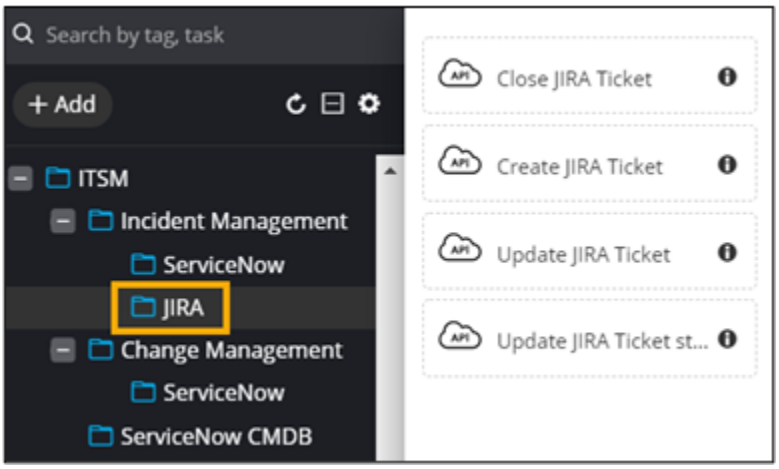
- ServiceNow
- Jira
- Incident Management
- Change Management
- ServiceNow CMDB

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



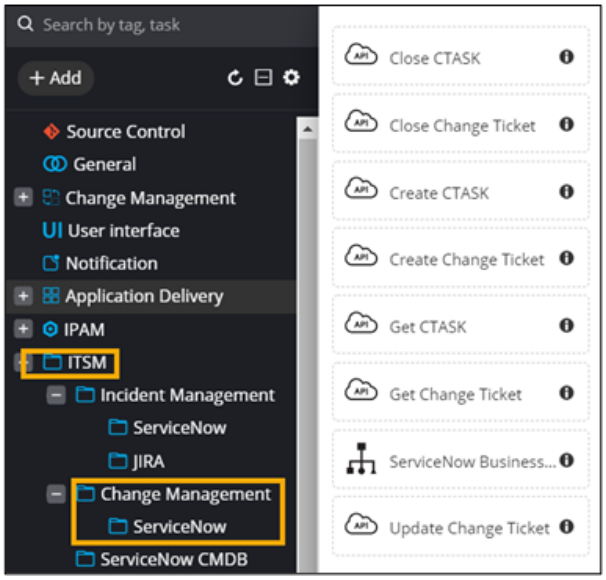


Note: For more information on using these tasks in a workflow, click here.

Change Management

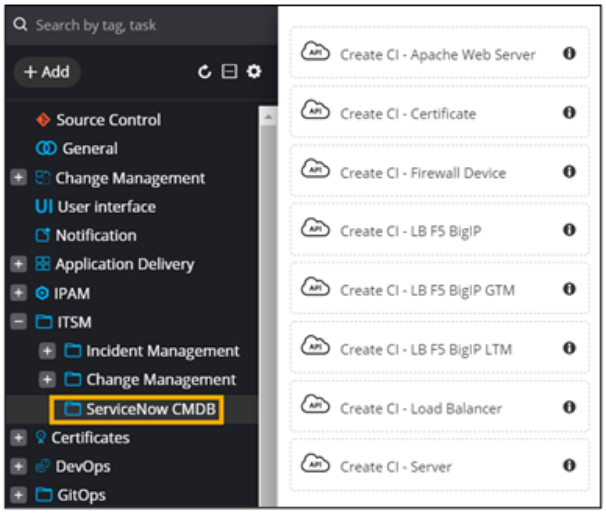
The tasks available in the **Change Management** folder allow you to design workflows for creating, updating and closing change tickets on ServiceNow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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.



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.

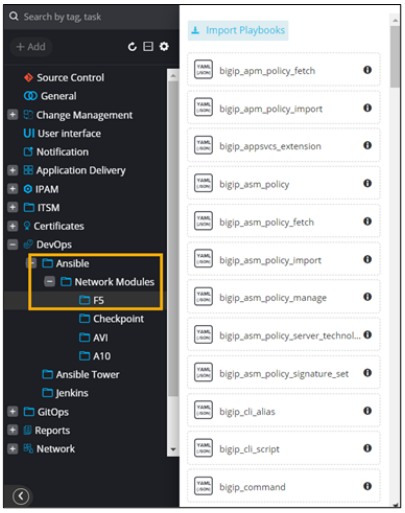
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Ansible
- Ansible Tower
- Jenkins

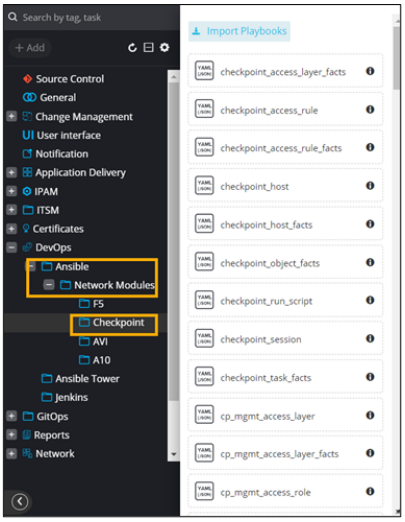
Ansible

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

- **F5**

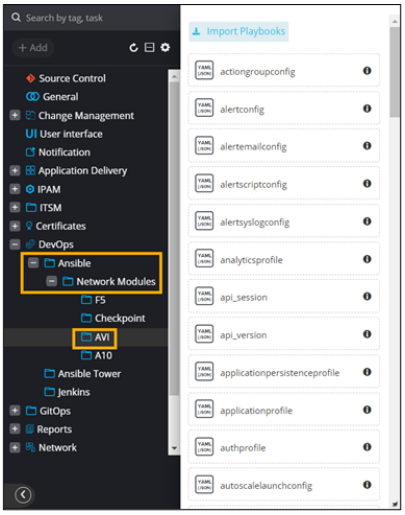


- **Checkpoint**

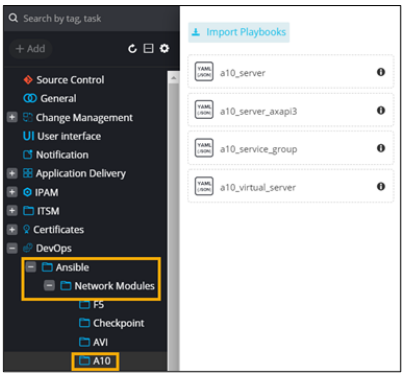


- **AVI**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



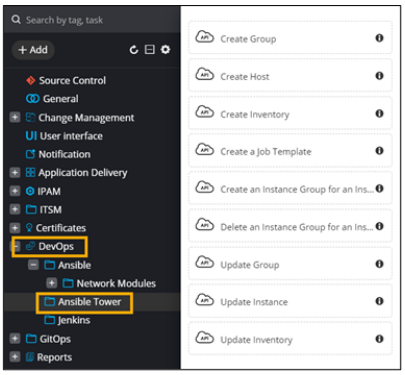
- **A10**



There is also a provision to import playbooks from Ansible.

Ansible Tower

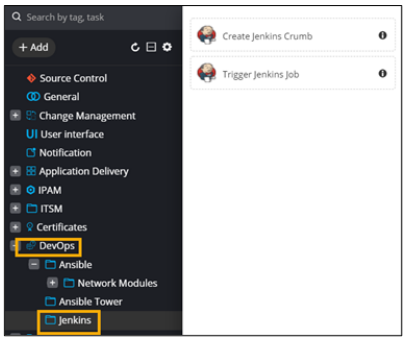
This folder consists of API tasks that can be used to create configurations on Ansible Tower and automate your tasks.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Jenkins

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

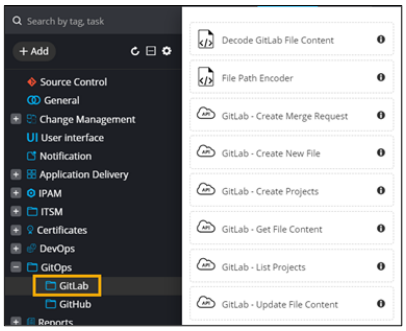


GitOps Integration

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

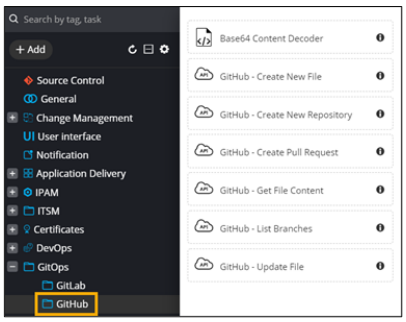
GitLab

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.



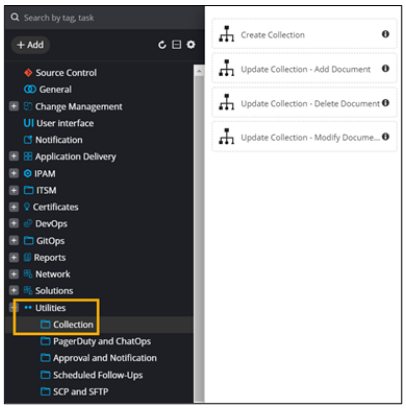
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.

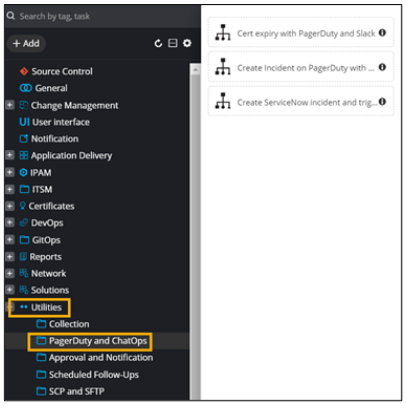
Collection

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.



Pagerduty and ChatOps

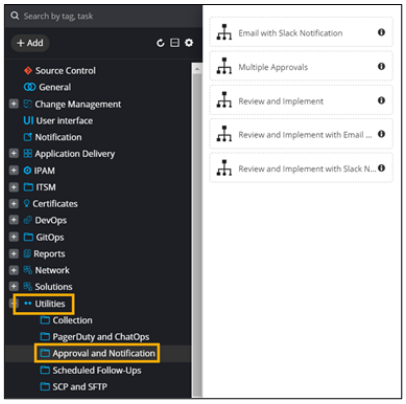
This folder contains prebuilt workflows for triggering alerts and notifications on PagerDuty, Slack, and other ChatOps after task execution.



Approval and Notification

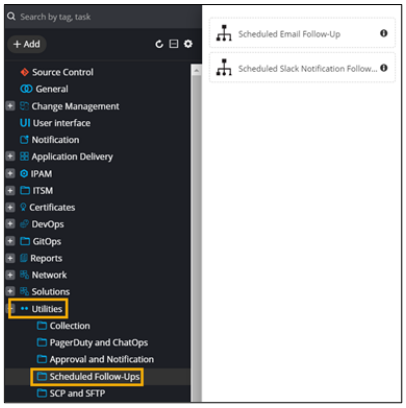
This folder contains prebuilt workflows for sending approvals and notifications via email. These workflows can be used for seeking multiple approvals for task execution (review and implementation) and sending email notifications once a task is complete.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



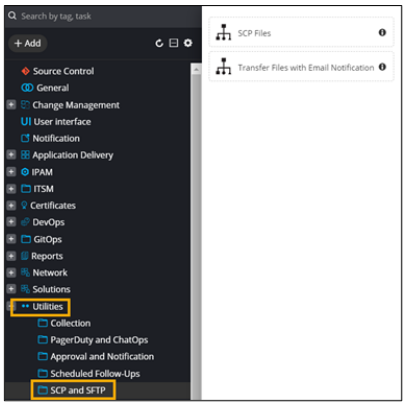
Scheduled Follow-Ups

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



SCP and SFTP

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



Copyright © 2025 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:

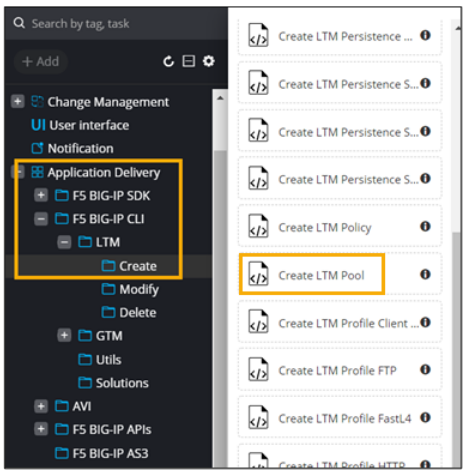
- AVI
- Bluecat
- F5
- Infoblox.

The pathway for the AppViewX folder is:

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

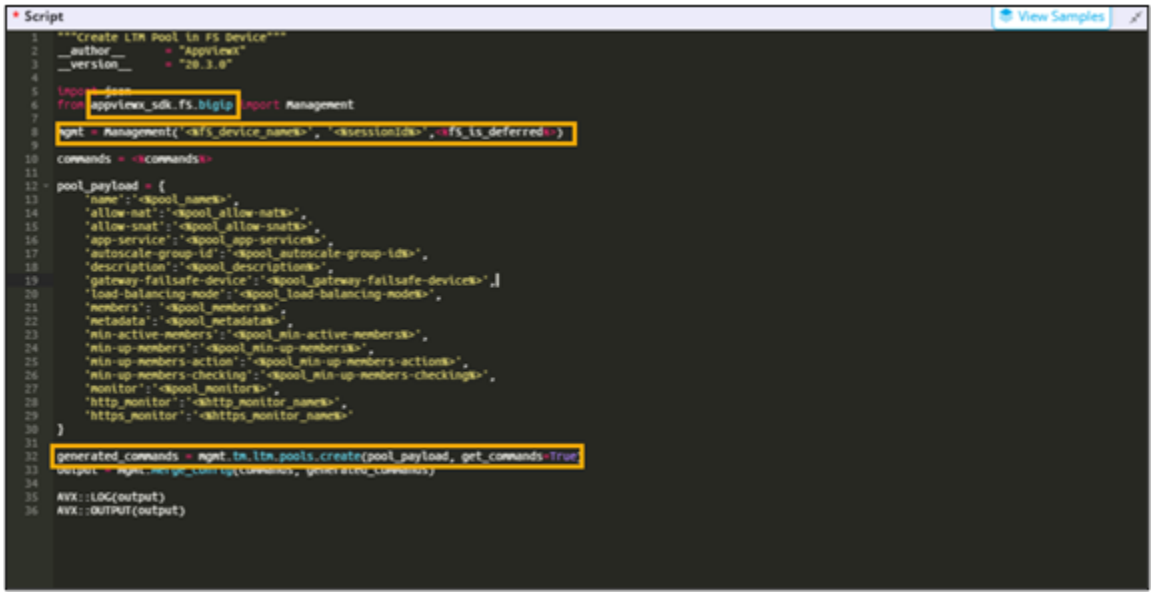
```
-bash-4.2$ pwd
/home/appviewx/appviewx/appviewx_dependencies/vw/dependencies
-bash-4.2$ ls -ltr | grep appviewx_sdk
drwxrwxr-x  8 appviewx appviewx      125 Jun 29 02:47 appviewx_sdk
-bash-4.2$ ls -ltr appviewx_sdk/
total 4
-rw-rw-r-- 1 appviewx appviewx 249 Jun  9 07:12 README.md
-rw-rw-r-- 1 appviewx 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 new workflow. a workflow.
2. In the design workspace, from the General section, select **Application Delivery** > **F5 BIG-IP CLI** > **LTM** > **Create**.
3. From the **Create** folder, drag and drop the **Create LTM Pool** task.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

AppViewX SDK has been used in the script along with a command for creating an LTM pool.



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

Workflow Studio

- Accessing the Workflow Page
- Workflow Actions
- Designing a Workflow
- Accessing Solutions
- Accessing the Store
- Importing Workflows
- Collection
- Hooks Inventory
- Magic Variable
- Helper Scripts
- Regex Library
- Request Settings
- Provisioning for Workflow Tasks

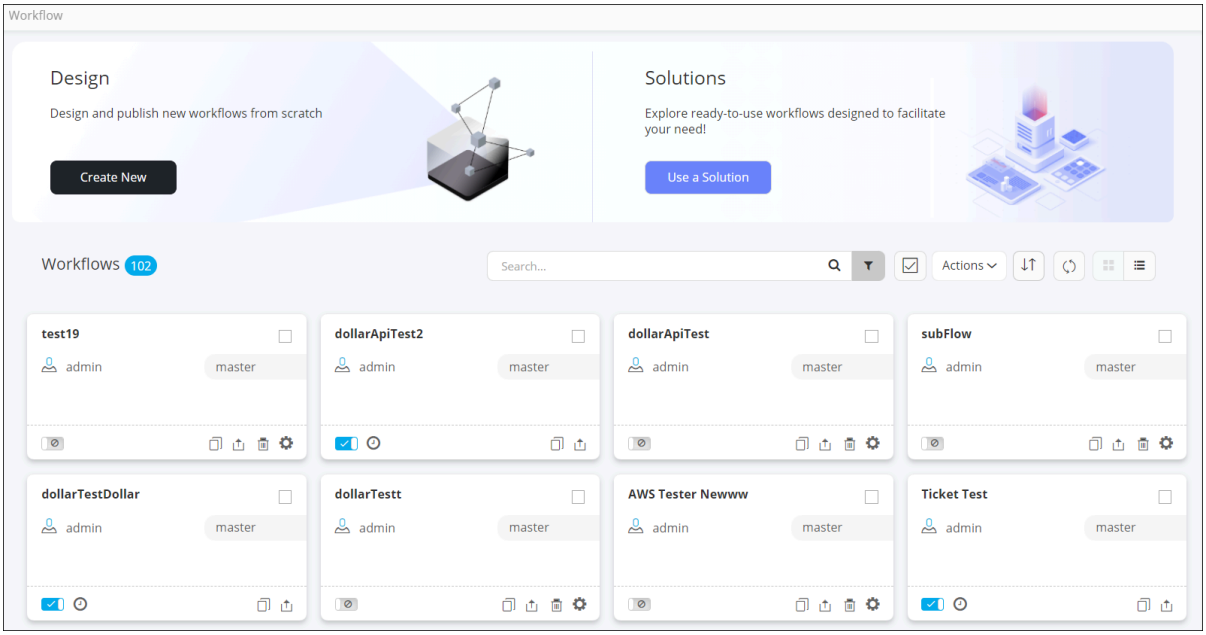
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Source Control
- Integration

Accessing the Workflow Page

New Menu	Old Menu
1. From the main navigation menu, select **Automation**. 2. From the navigation pane on the left, under **WORKFLOW**, select **Design**. The **Workflow** page is displayed.	From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed.
Note: For more information on how to switch between menus, click here.	

The **Workflow** page gives a consolidated view of the important aspects of the Visual Workflow platform, and inventory of all the custom-built workflows, and the actions that can be performed from the Workflow studio console.



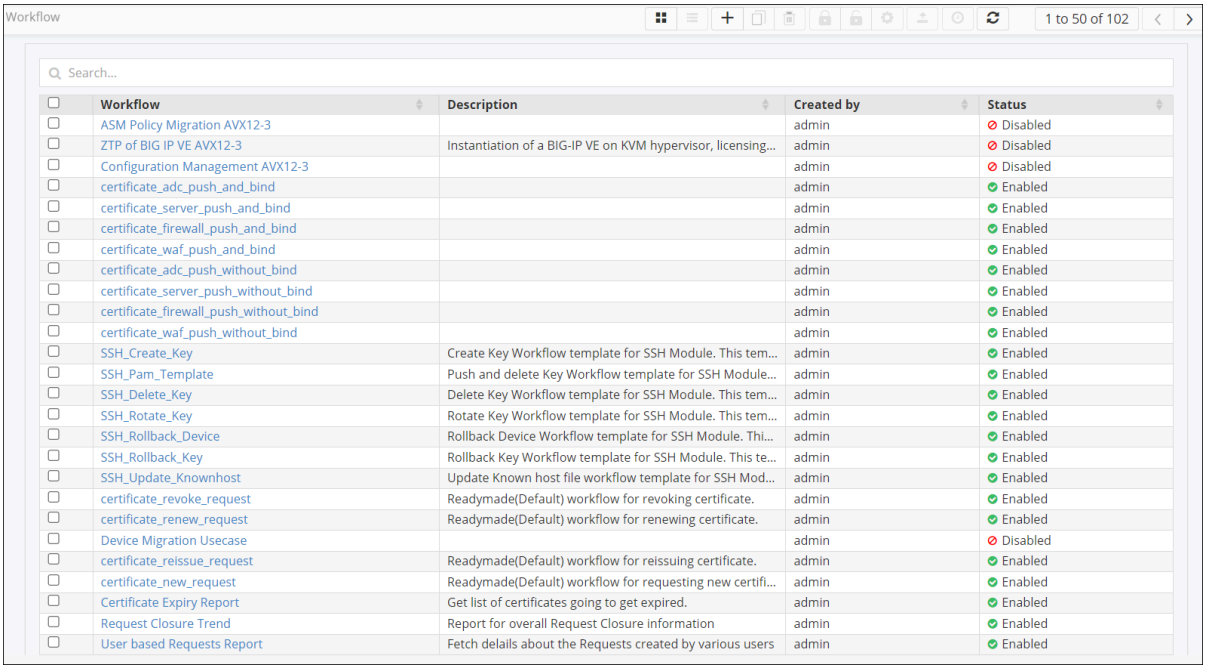
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Various options available on the Workflow page when the workflows are in Card view

Option	Description
Design	Allows you to create a new workflow from scratch. **Note:** For more information, click here.
Solutions	Allows you to access the ready-to-use OOB Solutions and FAQs. **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.
	Allows you to select all workflows displayed on the page with a single click.
Actions	Allows you to perform the following actions on workflows: • **Export** • **Enable** • **Disable** • **Delete** **Note:** For more information, click here.
	Allows you to rearrange the workflows as: • Ascending • Descending • Recently created
	Refreshes the page.
	Displays the workflows in a card view.
	Displays the workflows in a list view.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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



Various options available on the Workflow page when the workflows are in List view

Option	Description
Search bar	Allows you to search for a particular workflow by typing in the keyword.
	Displays the workflows in a card view.
	Displays the workflows in a list view.
	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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Various options available on the Workflow page when the workflows are in List view (continued)

Option	Description
	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:** Only enabled workflows can be scheduled. For more information, click here.
	Refreshes the page.

Workflow Actions

The **Actions** menu on the **Workflow** page (Card view) allows you to perform various actions on the workflows such as export, enable, disable, and delete.

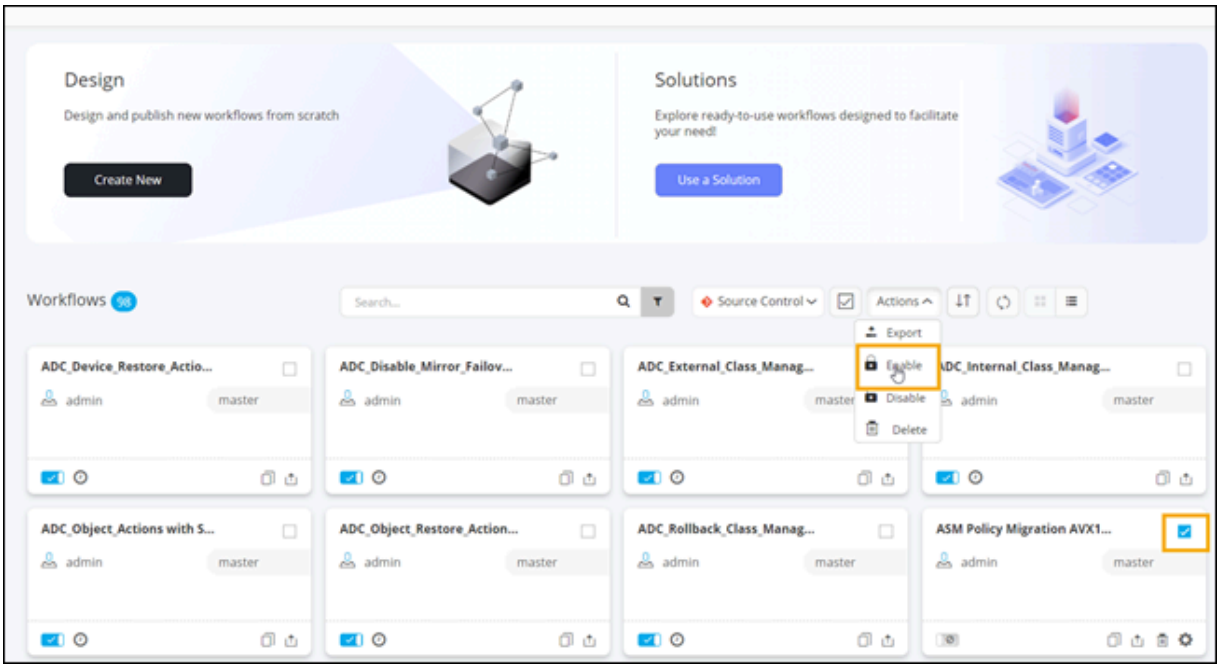
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: In the **List view** of the Workflow inventory, these actions are available in the command bar on the top right corner of the page.

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

Enabling a Workflow

1. Go to the Workflow page.
2. On the **Workflow** page (Card view), select the workflow to be enabled.
3. To enable the selected workflow, from the **Actions** dropdown menu, select **Enable**.



4. Click **Yes** in the **Confirmation** pop-up window.
   The workflow is enabled.

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

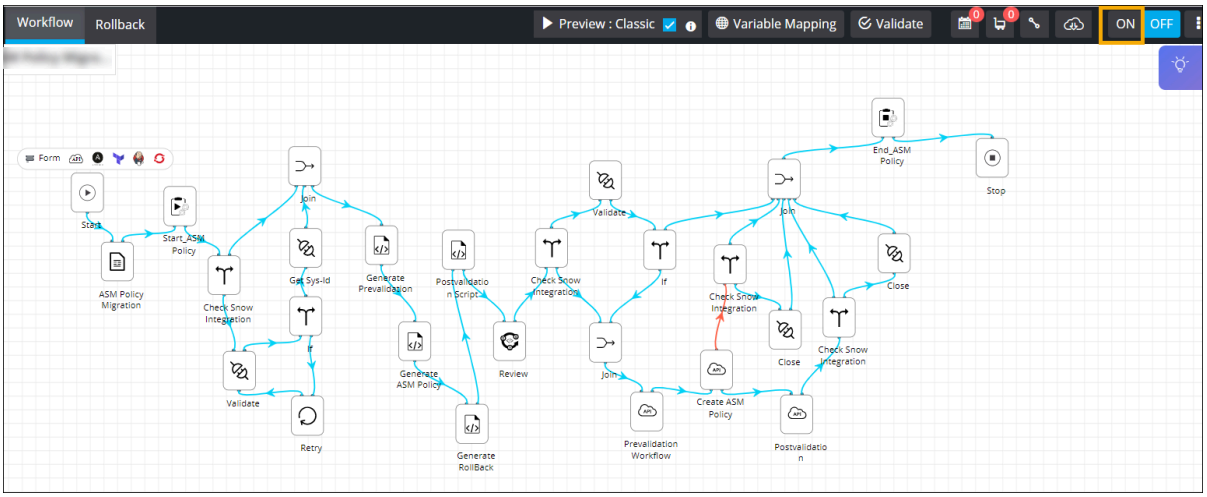
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

5. To enable multiple workflows, you can either individually select the workflows to be enabled or click **(Selection)** icon to select all workflows displayed on the page and click **Enable** in the **Actions** dropdown menu.
6. To enable a workflow, when the **Workflow** page is in **List view**, select the workflow(s) to be enabled and click **(Enable)** icon.

• Enabling a Workflow in Design Mode

Enabling a Workflow in Design Mode

1. In the workflow design space, from the right upper corner of the screen, click **ON**.



2. In the **Confirmation** pop-up window, click **Validate & Enable**.

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

The workflow is enabled.

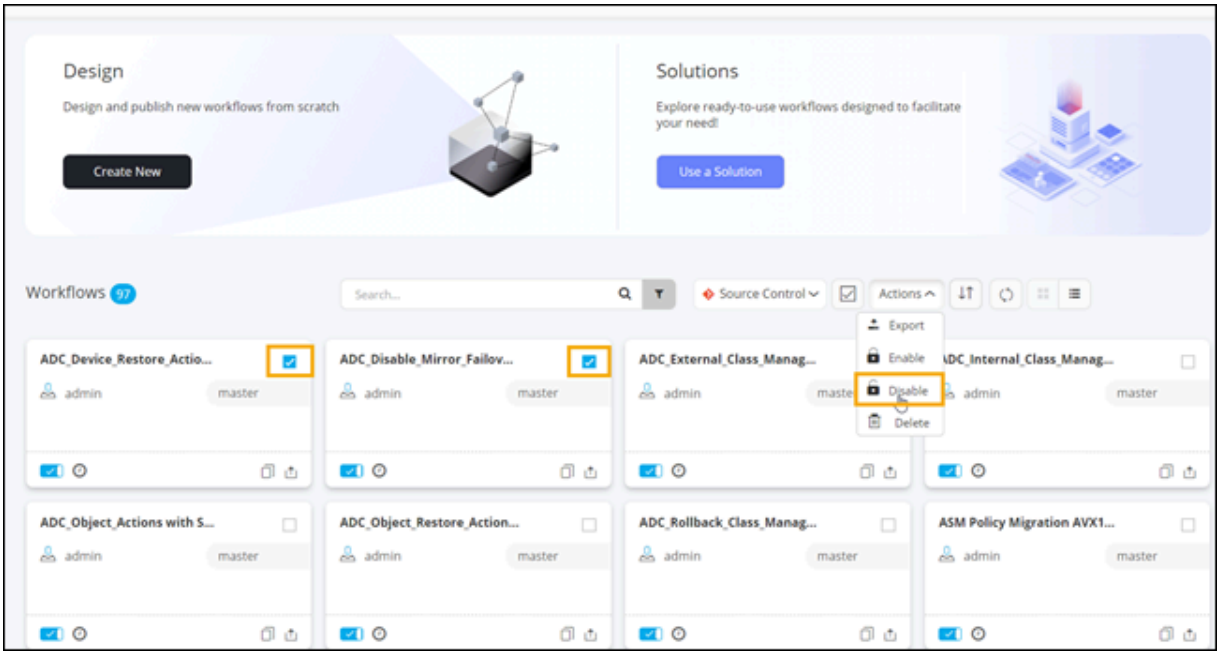
> **Important:** You cannot modify a workflow when it is in enabled state.

Disabling a Workflow

1. Go to the Workflow page.
2. On the **Workflow** page (Card view), select the workflow to be disabled.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. To disable the selected workflow, from the **Actions** dropdown menu, select **Disable**.

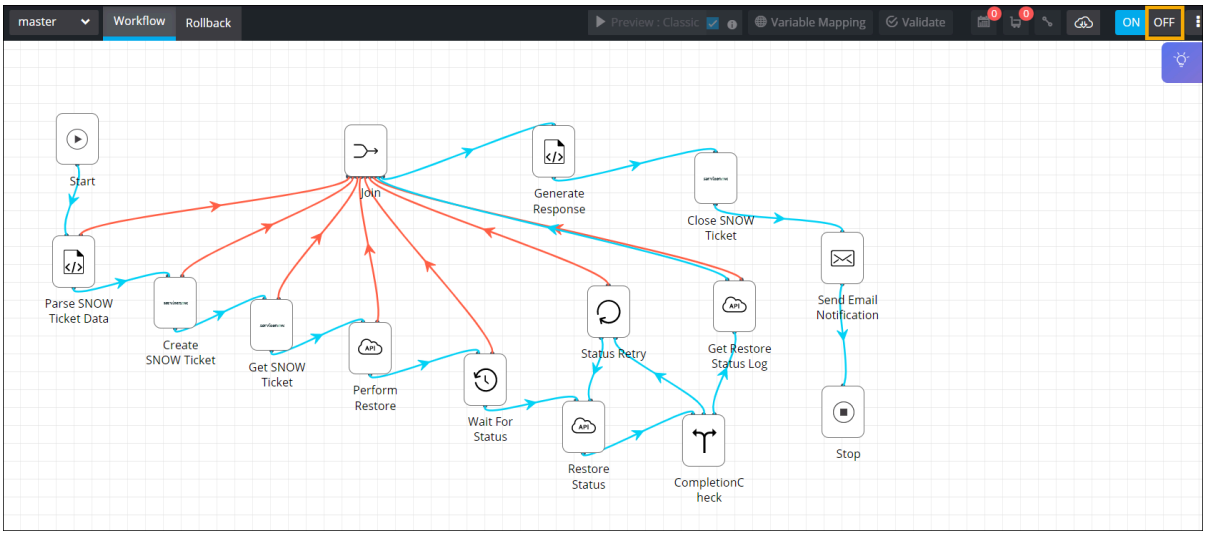


4. Click **Yes** in the **Confirmation** pop-up window.
   The workflow is disabled.

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

5. To disable multiple workflows, you can either select the workflows to be disabled individually or click **(Selection)** icon to select all workflows displayed on the page and click **Disable** in the **Actions** dropdown menu.
6. To disable a workflow in the design space, from the upper right corner of the screen, click **OFF**.

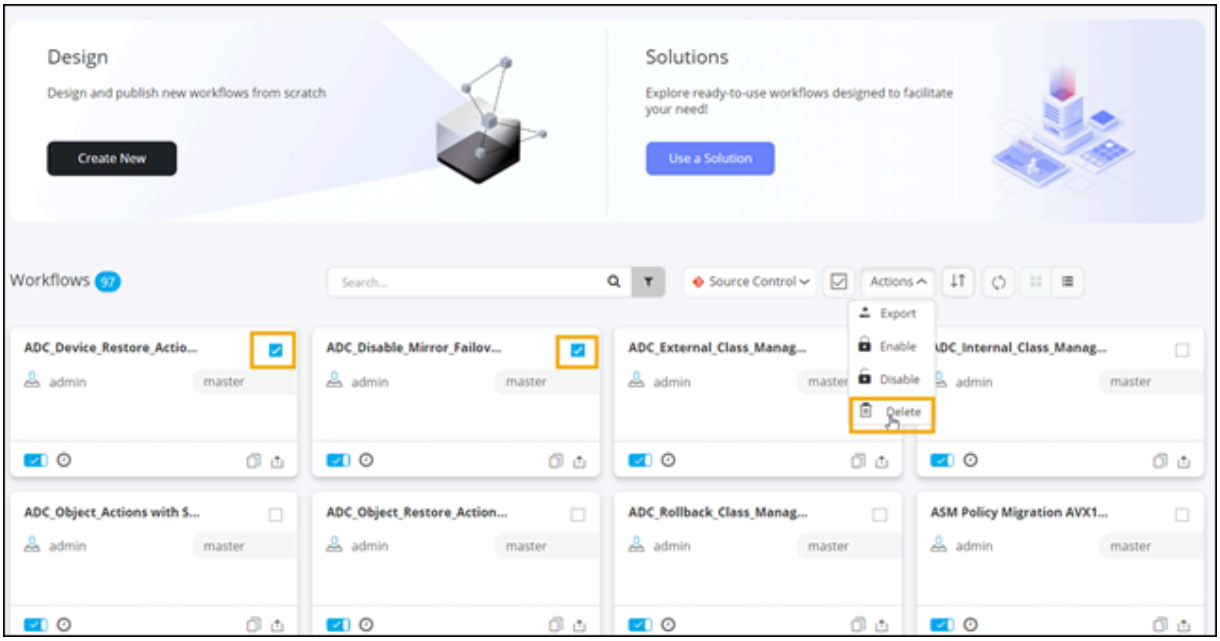
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

Deleting a Workflow

1. Go to the Workflow page.
2. On the **Workflow** page (Card view), disable the workflow to be deleted.
3. Select the workflow.
4. To delete the workflow, from the **Actions** dropdown menu, select **Delete**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

5. To delete multiple workflows, either select them individually or click **(Selection)** icon to select all workflows displayed on the page and click **Delete** in the **Actions** dropdown menu.

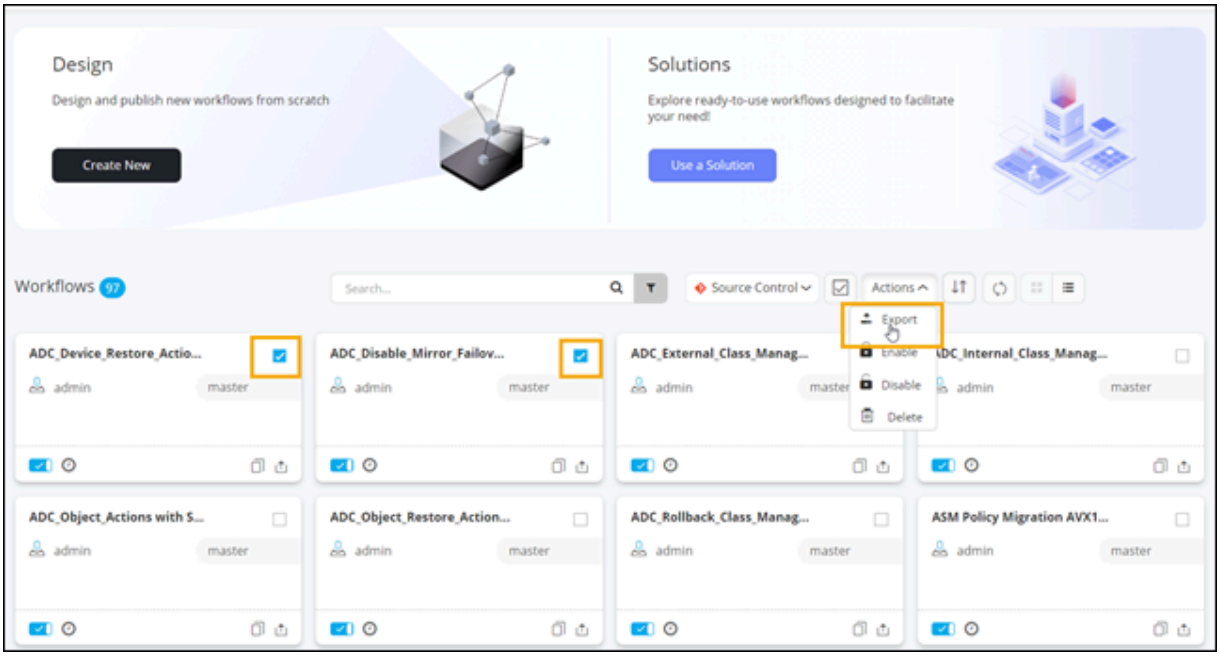
> **Note:** Ensure that the workflows to be deleted are disabled.

6. To delete a workflow when the **Workflow** page is in **List view**, select the workflow(s) to be deleted and click **(Delete)** icon.

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

Exporting a Workflow

1. Go to the Workflow page.
2. On the **Workflow** page (Card view), select the workflow(s) to be exported.
3. To export the workflow(s), from the **Actions** dropdown menu, select **Export**.



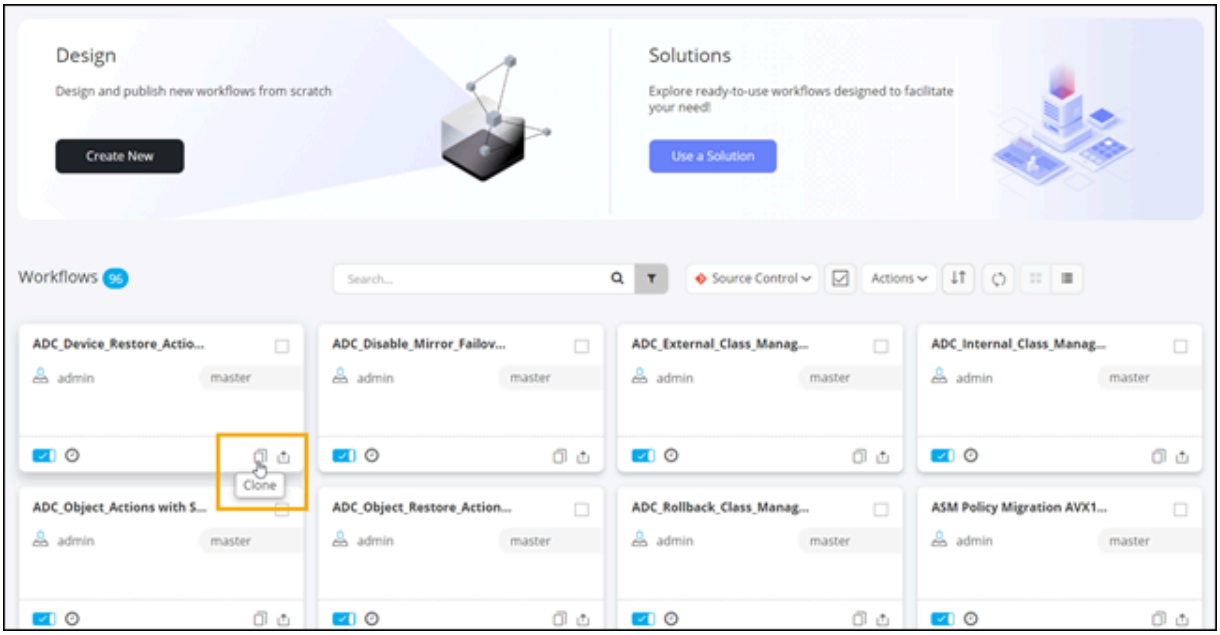
4. Click **Ok** in the **Export** window.
   The selected workflow is saved to your machine as a .zip file.
5. To export a flow, you can also click **(Export)** icon on the workflow card.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. To export multiple workflows, you can either select them individually or click **(Selection)** icon to select all workflows displayed on the page and click **Export** in the **Actions** dropdown menu.
7. To export a workflow, when the **Workflow** page is in **List view**, select the workflow(s) to be enabled and click **(Export)** icon.

Cloning a Workflow

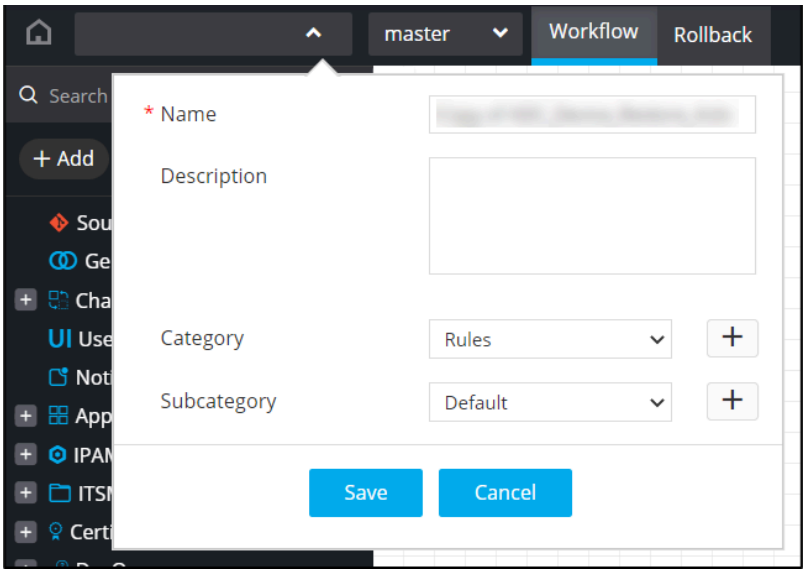
1. Go to the Workflow page.
2. On the **Workflow** page (Card view), click **(Clone)** icon on the workflow to be cloned.



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

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

Note: To clone a workflow when the **Workflow** page is in **List view**, select the workflow to be cloned and click **(Clone)** icon.

Designing a Workflow

The Design section in the Workflow studio allows you to create a new workflow or modify an existing workflow. 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.
- 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).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

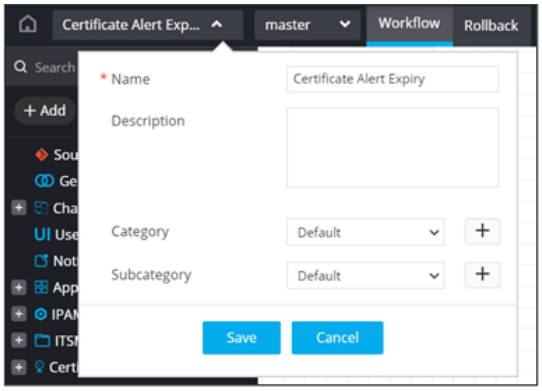
- Provision to preview a workflow task.
- Provision to preview a workflow/nested workflow.
- Provision to reuse and/or import workflow and workflow task(s).

To design a new workflow:

1. Go to the Workflow page.
2. On the **Workflow** page (Card view), click **Create New**.

> **Tip:** When the **Workflow** inventory page is in List View, you can click + **(Add)** icon in the command bar to design a new workflow.

3. In the workflow design space, enter or select the requested field information.



Field descriptions for workflow design space

Field	Description
***Name**	Enter a name for the workflow.
Description	Enter a description for the workflow.
Category	Select a category from the options displayed in the dropdown. **Note:** You can create a new category by clicking + **(Add)** icon.
Subcategory	Select a subcategory from the options displayed in the dropdown. **Note:** You can create a new subcategory by clicking + **(Add)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
** : Mandatory fields*	

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 getting started quickly with building workflows, click here.

• Modifying a Workflow

Modifying a Workflow

You can modify a pre-existing workflow to suit user-specific requirements.

> **Note:** Before modifying a workflow, ensure that it is disabled.

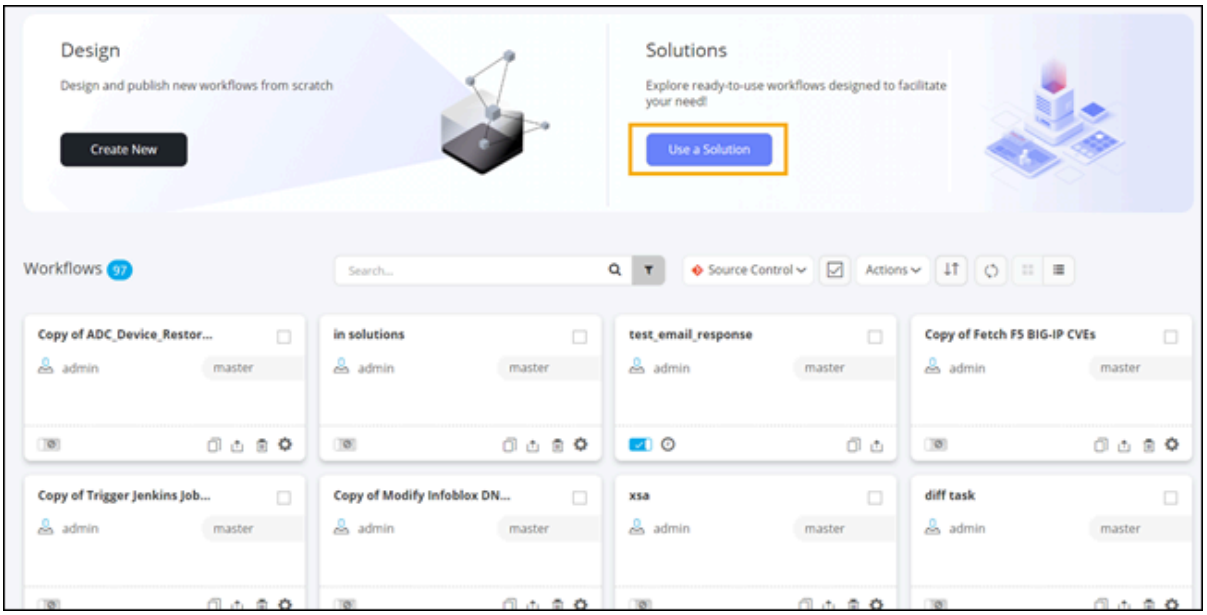
1. Go to the Workflow page.
2. On the **Workflow** page, click on the workflow that you want to modify.
   The workflow opens in the design space.
3. Modify the task(s) in the workflow as required

Accessing Solutions

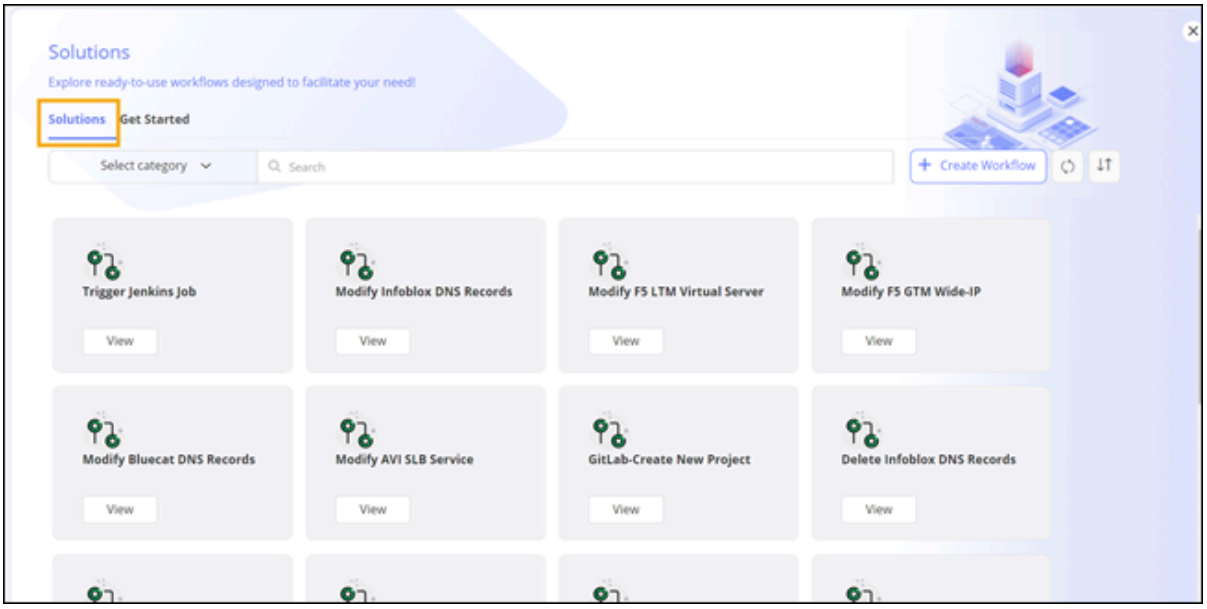
This section allows you to easily access the out of the box (OOB) solutions and FAQs from the Workflow page. You can use any of these solutions to build a workflow quickly and easily.

1. Go to the Workflow page.
2. On the **Workflow** page, click **Use a Solution**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **Solutions** page is displayed with the **Solutions** tab open by default.



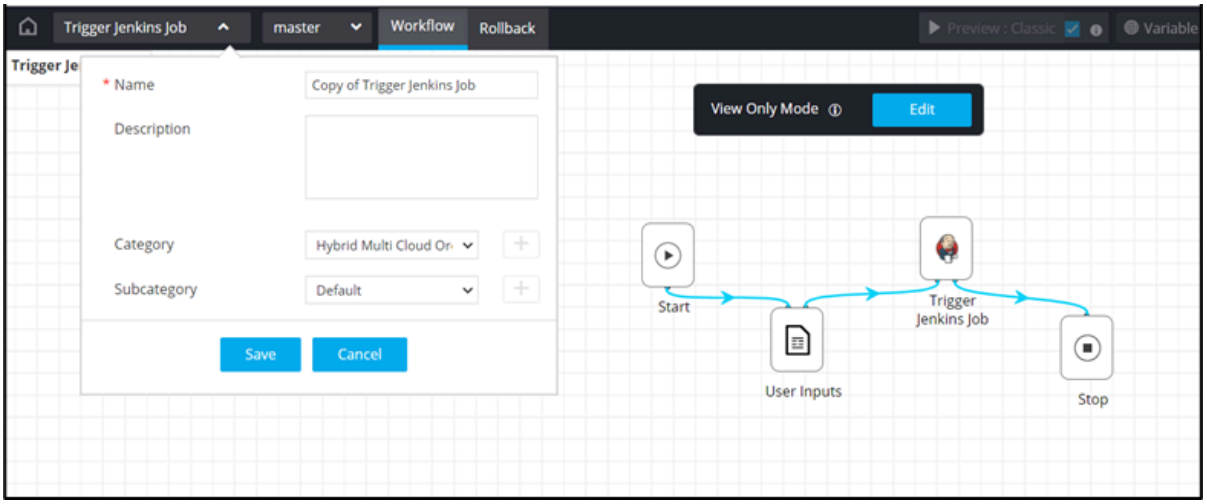
Options available on Solutions tab

Option	Description
Solutions	The **Solutions** tab displays the ready-to-use **solutions** available in the Workflow Studio.
Get Started	The **Get Started** tab displays the **FAQs** available in the Workflow Studio.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
Select Category	Allows you to filter solutions according to the category selected in the dropdown list.
Search	Allows you to search for a solution by typing the keyword(s).
Create Workflow	Allows you to create a workflow.
	Refreshes the page.
	Allows you to rearrange the workflows as • Ascending • Descending • Recently created.
	Allows you to exit the **Solutions** page and return to the **Workflow** page (Card view).

3. To open a ready-to-use solution in the design space, click **View** on the corresponding card.
   The workflow opens in a **View Only Mode** in the design workspace.
4. To use this solution, click **Edit**.
5. Enter a new **Name** for the workflow, if required.



6. Click **Save**.
   The workflow can now be modified as required.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

> **Note:** You can also access solutions from within a workflow open in the design canvas by clicking icon in the top right corner.

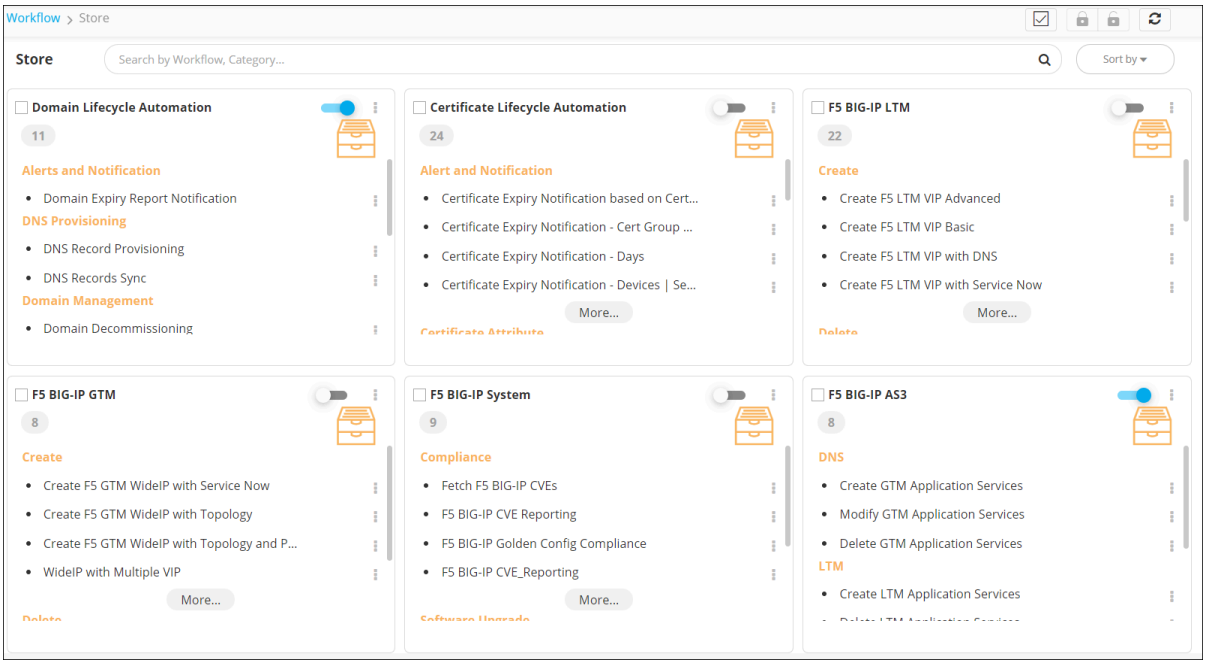
7. To access quick out of the box solutions that can be consumed within workflows, on the Solutions page, click **Get Started** tab.
   The **Get Started** tab displays these OOB solutions.

Accessing the Store

New Menu	Old Menu
In the **Automation** module, from the navigation pane on the left, under **WORKFLOW**, select **Store**. The **Workflow** > **Store** page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Store**. The **Workflow** > **Store** page is displayed.
> **Note:** For more information on how to switch between menus, click here.	

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 under one catalog. The **Store** can be accessed under **WORKFLOW** section.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Actions you can perform on the Store page

Action	Description
Search bar	Allows you to search for a specific workflow in the Store.
	Allows you to select all workflow catalogs on the Store page.
	Allows you to enable all the workflows in a selected workflow catalog(s). **Note:** This action is enabled only when you select a workflow catalog(s).
	Allows you to disable all the workflows in a selected workflow catalog(s). **Note:** This action is enabled only when you select a workflow catalog(s).
	Refreshes the page
Sort by	Allows you to sort the catalogs.

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

- Enabling/Disabling OOB Workflow Catalogs
- Enabling/Disabling individual OOB Workflows
- Cloning an OOB Workflow

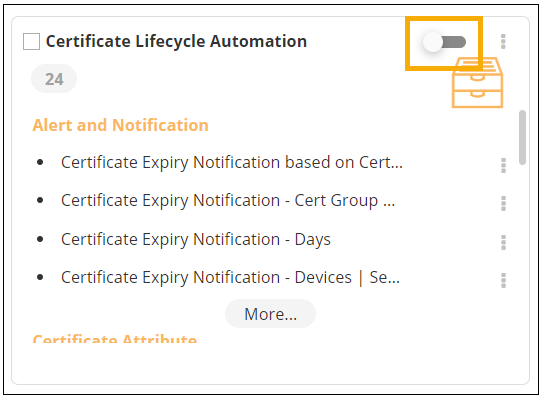
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Scheduling an OOB Workflow
- Configuring RBAC for OOB Workflows
- Sorting the Workflow Catalogs

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, all the workflows categorized under the catalog will not be available under **Service Requests** and therefore cannot be scheduled or triggered.

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



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

> **Tip:** You can also enable or disable multiple catalogs by selecting the checkbox for the catalog(s) and clicking **(Enable)** icon to enable or **(Disable)** icon to disable the catalog(s).

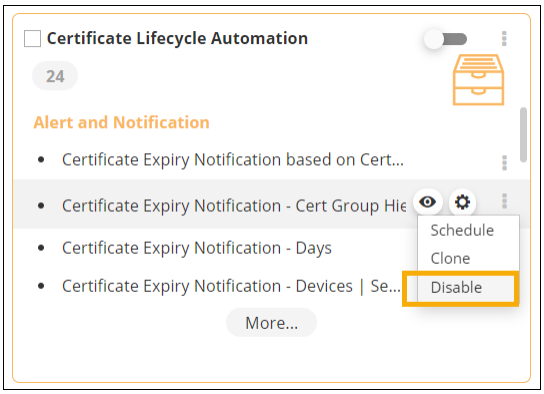
Enabling/Disabling individual OOB Workflows

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:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Go to the Store.
2. On the **Workflow > Store** page, hover your mouse over a workflow name, and then click **(Dots)** icon.
3. From the available options, click **Disable**.



The workflow is disabled.

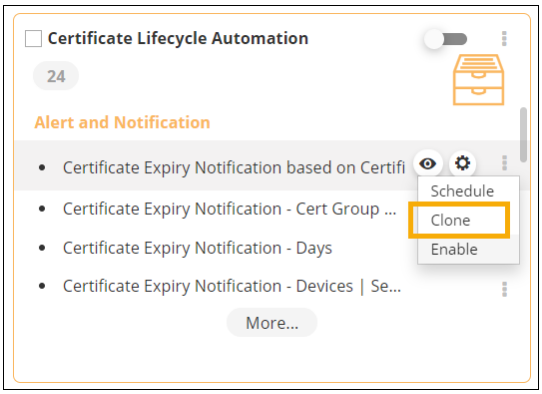
4. To enable a workflow, on the **Workflow > Store** page, hover your mouse over the workflow to be enabled, and then click **(Dots)** icon.
5. From the options available, click **Enable**.
   The workflow is enabled.

Cloning an OOB Workflow

To clone an individual OOB workflow inside a workflow catalog:

1. Go to the Store.
2. On the **Workflow > Store** page, hover your mouse over the workflow to be cloned, and then click **(Dots)** icon.
3. From the options available, click **Clone**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

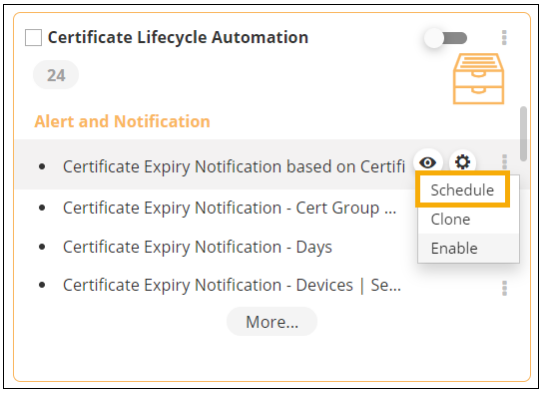


A copy of the cloned workflow is added to the **Workflow** inventory page.

Scheduling an OOB Workflow

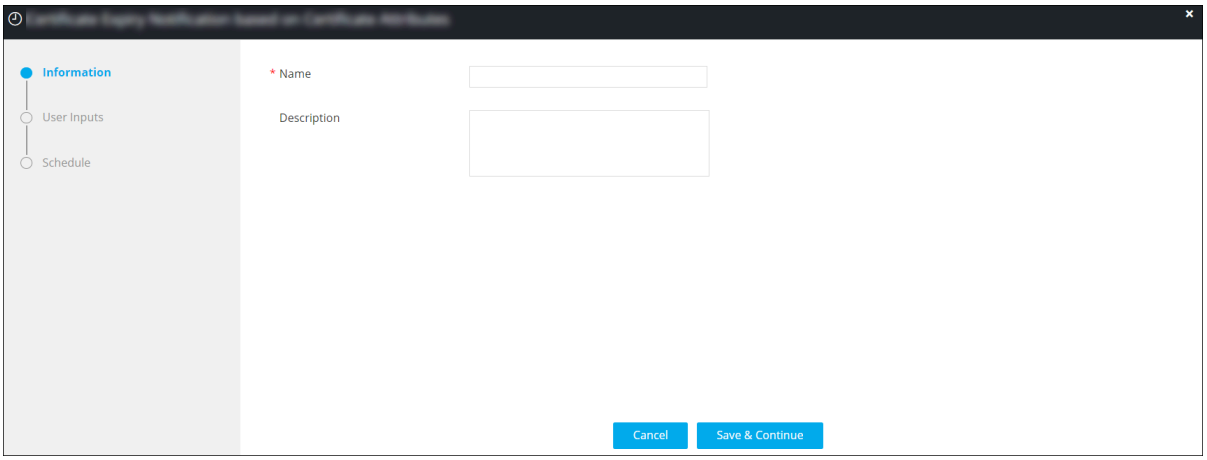
To schedule an OOB workflow from the Store:

1. Go to the Store.
2. On the **Workflow** > **Store** page, hover your mouse over the workflow you want to schedule, and then click **(Dots)** icon.
3. From the options available, click **Schedule**.



4. In the pop-up window that is displayed, under required field information for scheduling this workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

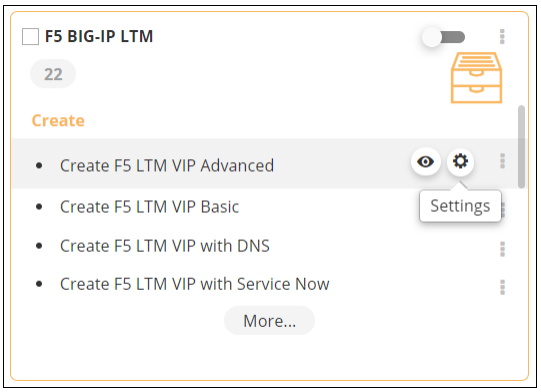


5. Click **Schedule**.
   The scheduled workflow is added to the **Scheduled jobs** on the Request :: Schedule jobs page.

Configuring RBAC for OOB Workflows

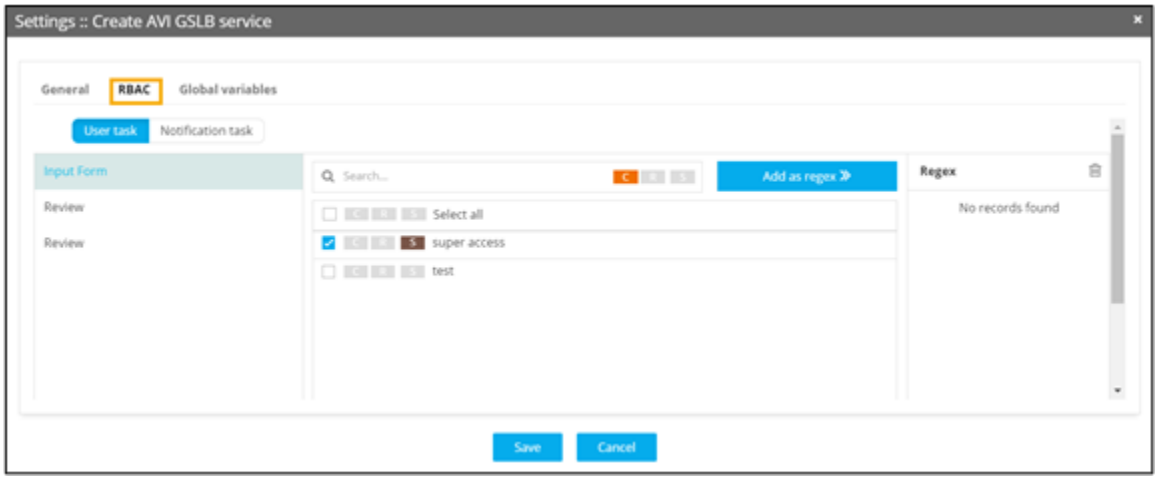
To configure RBAC settings for a particular OOB workflow:

1. Go to the Store.
2. On the **Workflow > Store** page, hover your mouse over the workflow, and then click **(Settings)** icon.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

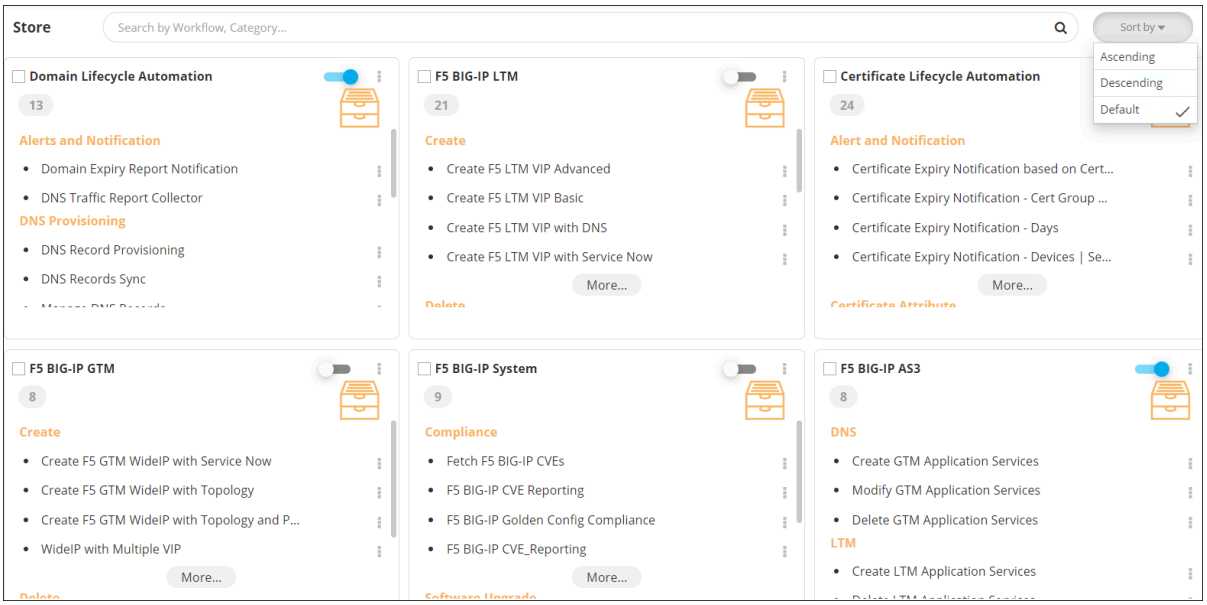


When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.

Sorting the Workflow Catalogs

You can sort the workflow catalogs in ascending or descending order.

1. On the Store page, from the top right corner of the page, click **Sort by**.



2. From the **Sort by** dropdown menu, select how you want to arrange the workflow catalogs. The options available are:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Ascending**: All workflow catalogs will be sorted in an ascending order as per the catalog name.
- **Descending**: All workflow catalogs will be sorted in a descending order as per the catalog name.
- **Default**: Workflows catalogs will be displayed in any random order.

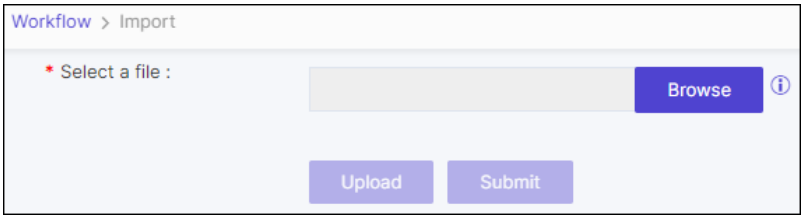
Importing Workflows

You can import workflow(s) from one environment to the other.

To import a workflow:

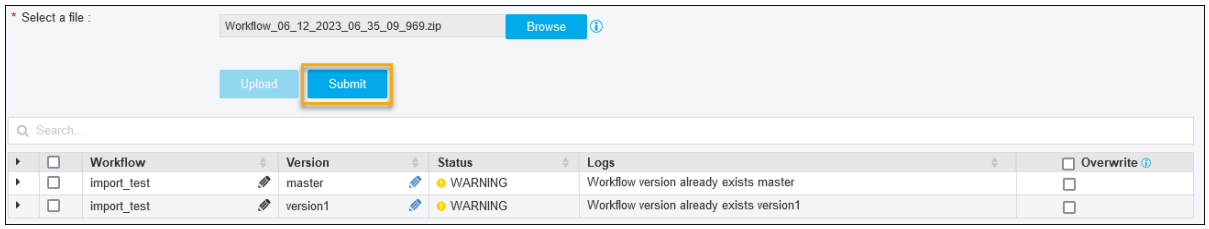
New Menu	Old Menu
In the Automationmodule, from the navigation pane on the left, under **WORKFLOW**, select **Import**. The **Workflow** > **Import** page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Import**. The **Workflow** > **Import** page is displayed.
Note: For more information on how to switch between menus, click here.	

1. On the **Workflow** > **Import** page, to select the file to be imported, click **Browse**.

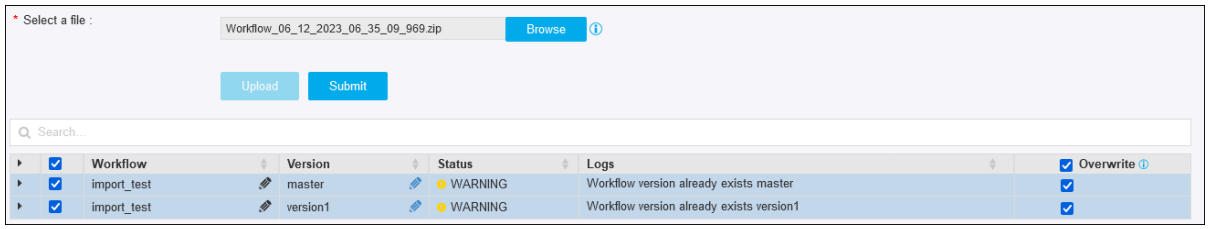


2. Select the file from the file explorer on your computer.
3. To upload the selected .zip file, click **Upload**.
4. Select the workflow, and then click **Submit**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

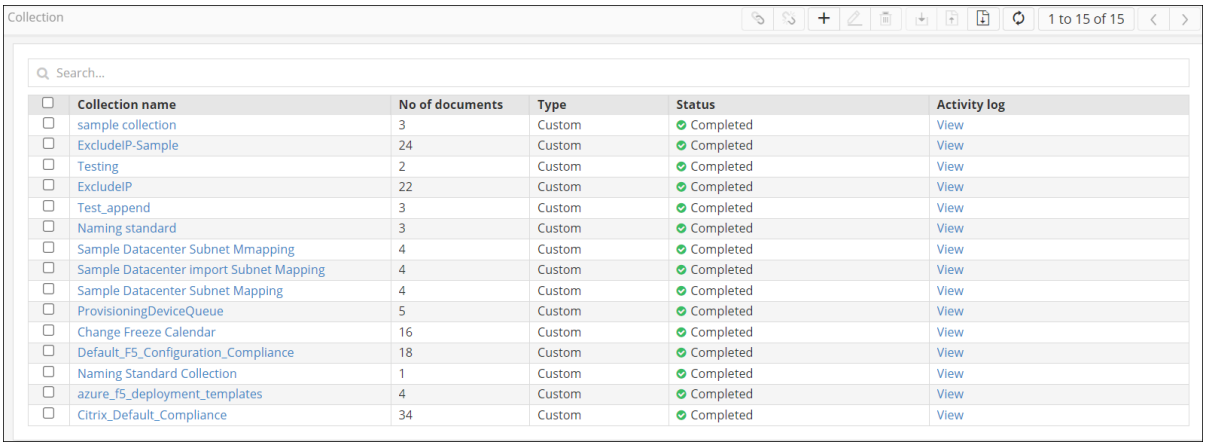


Warning: If you are importing a workflow that already exists, the workflow status will show a Warning. You need to either edit the workflow name or the version to be able to import this workflow or you can choose the overwrite option to replace its existing content and the RBAC permission for the specific task will remain same with the existing workflow.



Collection

Collection is a placeholder for storing custom data which can be used in other workflows, reports, and service catalogs. Data generated from one workflow can be stored as a collection and used as a part of another workflow through APIs or Hooks.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Options available on the Collection page

Option	Description
	Allows you to enable the selected collection(s) for building a custom query. **Note:** This option is enabled only when you select a collection.
	Allows you to disable the selected collection(s) for building a custom query. **Note:** This option is enabled only when you select a collection.
	Allows you to create a new collection manually.
	Allows you to edit/update the selected collection(s). **Note:** This option is enabled only when you select a collection.
	Allows you to delete the selected collection(s). **Note**: This option is enabled only when you select a collection.
	Allows you to download the selected collection(s) as a .xls file. **Note:** This option is enabled only when you select a collection.
	Allows you to export the selected collection(s) as a .zip file. **Note:** This option is enabled only when you select a collection.
	Allows you to import collection(s) as .xls or .zip files.
	Refreshes the page.

- Creating Collections
- Importing Collections

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Modifying Collections
- Using Collections

Creating Collections

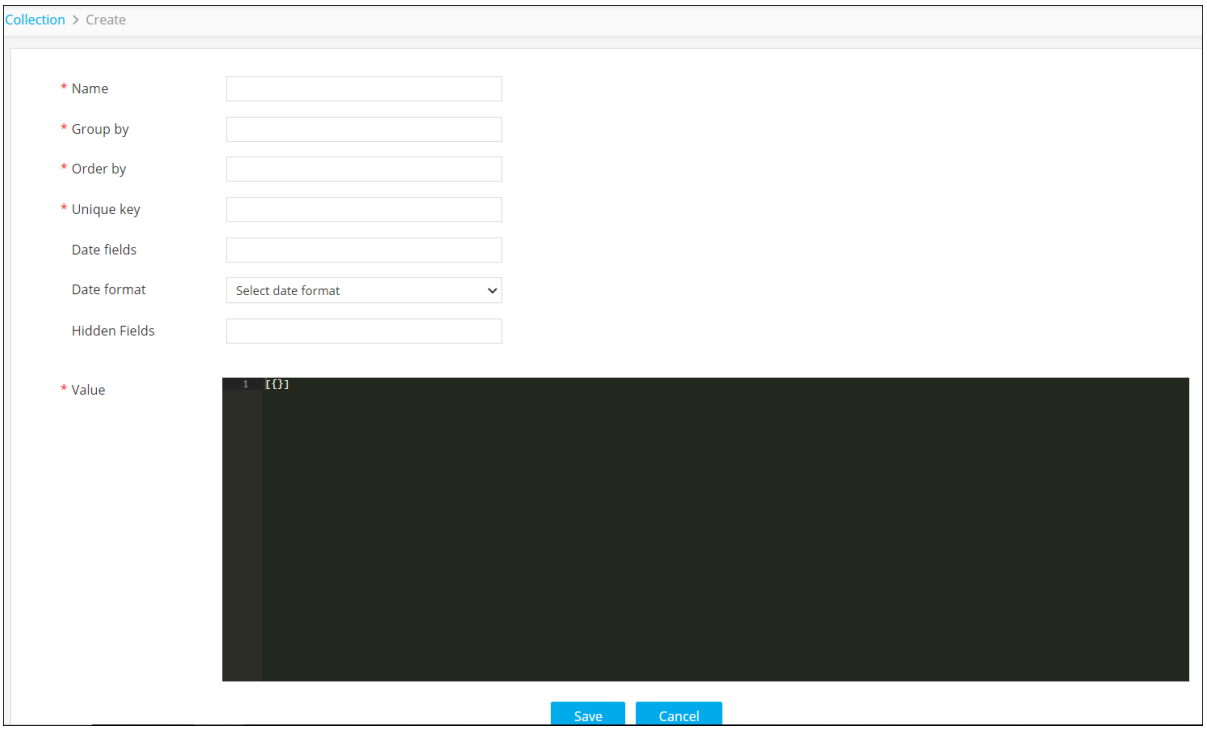
You can create a collection either by defining all the custom data for the collection from scratch or by using the sample collection file.

- Defining Custom Data
- Using Sample Collection File

Defining Custom Data

To create a new collection from scratch:

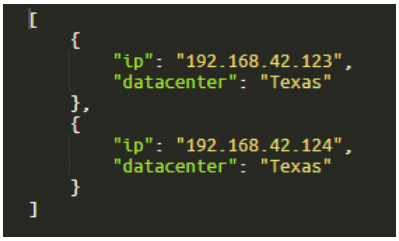
1. On the **Collection** page, from the command bar on the top right of the screen, click **(Add)** icon.
2. On the **Collection** > **Create** page, enter the required field information.



Field descriptions for create page

Field	Description
***Name**	Enter the name for the collection.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
	For example: ExcludeIP-Sample **Note:** The period (.) or dot symbol is allowed in the Name field. For example: ExcludeIP.Sample
***Group by**	Define how the collection will be grouped. For example: datacenter All the IPs will be grouped according to the datacenter.
***Order by**	Define how the data in the collection will be ordered. For example: ip The data in the collection will be ordered by the IPs
***Unique key**	Define the unique key or identifier for the collection.
Date fields	Define the date fields in the metadata, if required.
Date format	Define the date format, if required. For example: dd/mm/yyyy
Hidden Fields	Enter the hidden fields, if required.
***Value**	Define the data for the collection. For example: 
** : Mandatory fields*	

3. Click **Save**.

   The collection is added to the **Collection** page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Using Sample Collection File

To create a collection using the sample collection file:

1. On the **Collection** page, from the command bar on the top right of the screen, click **(Download)** icon.
2. On the **Collection** > **Import** page, to download a sample collection file, from the top right corner of the screen, click **(Download)** icon.
   A sample collection file <.xls> is downloaded to your computer.
3. Open the downloaded file and define the keys and metadata as required. A sample is shown here:

	A	B	C	D	E
1	MetaKeys	MetaValues	subnet	datacenter	zone
2	collectionNam	Collection Sample	192.168.33.0/26	NYE	Red
3	groupBy	zone	192.168.32.0/25	CBE	Green
4	orderBy	subnet			
5	uniqueKey	zone			
6					

4. Save the collection and import it into your instance.

Importing Collections

There are three ways to import a collection into your instance:

- Importing the Sample file
- Importing a <.xls> or <.zip> file
- Importing an Exported File

Importing the Sample file

For steps on how to import and use the sample collection file to create custom collections, click here.

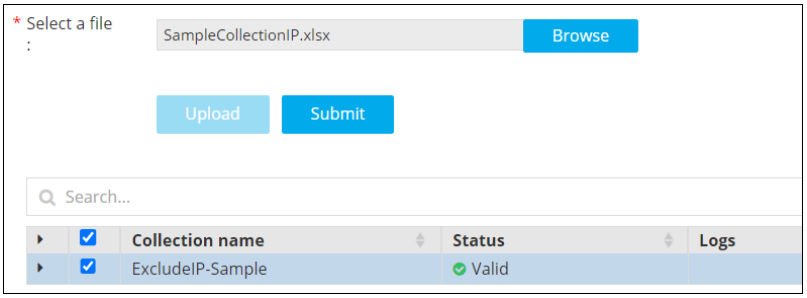
Importing a <.xls> or <.zip> file

To import a collection into AppViewX:

1. On the **Collection** page, from the command bar on the top right of the screen, click **(Download)** icon.
2. On the **Collection** > **Import** page, click **Browse**.

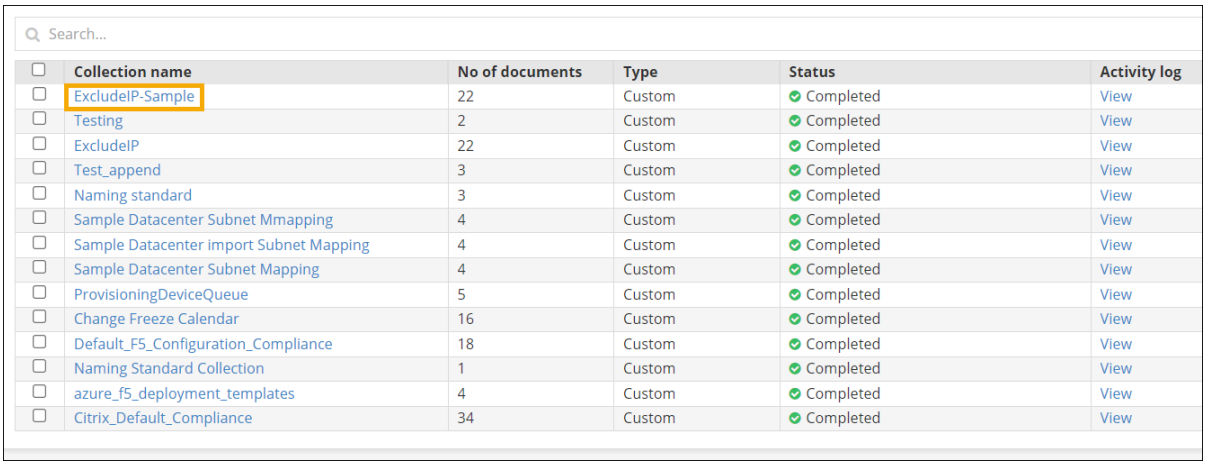
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Select the file to be imported into AppViewX and click **Upload**.

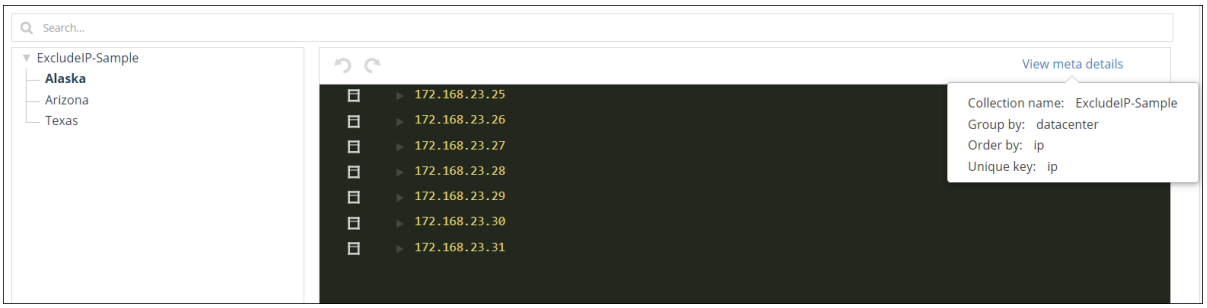


4. Select the checkbox next to the **Collection name**, and then click **Submit**.

   The imported collection is added to the **Collection** page.



5. Click **Collection name** and hover over **View meta details** for details about the collection.



- **Collection name:** This is the name of the collection. For example: **ExcludeIP-Sample**.
- **Group by:** You can define the parameter for grouping the data in the collection. For example: All the IPs in this collection will be grouped by datacenter - Alaska, Arizona, Texas, and so on. For more information, click here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Order by:** You can define how the data will be ordered in the collection For example: If you set the Order by field as ip, then all the IPs will be listed in order in the collection. For more information, click here.
- **Unique key:** This is the unique identifier for the data in the collection.

Importing an Exported File

You can export a collection from a different environment and import it into your environment or export a collection from your instance, modify it, and import it again.

1. To export a collection, on the **Collection** page, from the command bar, click **(Export)** icon. The file is saved to your laptop as a <.zip> file.

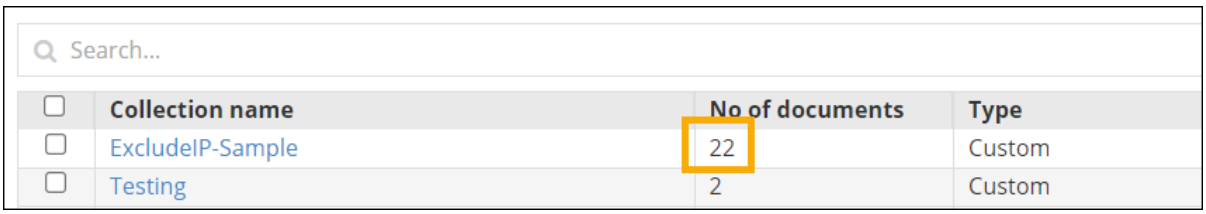
> **Note:** You can also download the collection as a **<.xls>** file by clicking from the command bar.

2. Modify this collection, if required and import it into your instance.

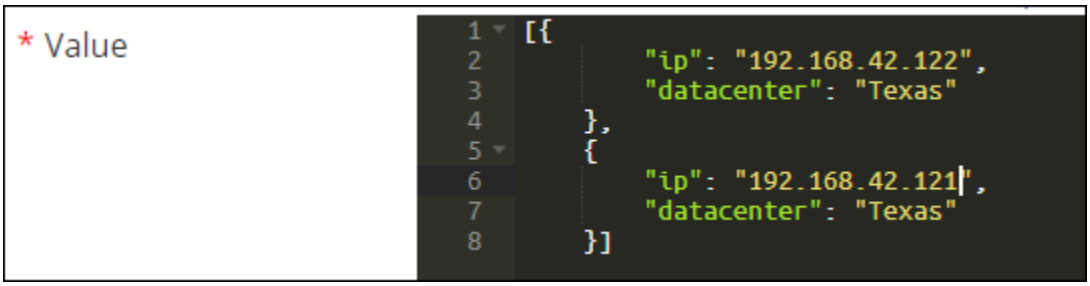
Modifying Collections

To update or modify data in a collection:

1. On the **Collection** page, select the collection to be modified. For example: ExcludeIP-Sample. This collection currently contains 22 documents.

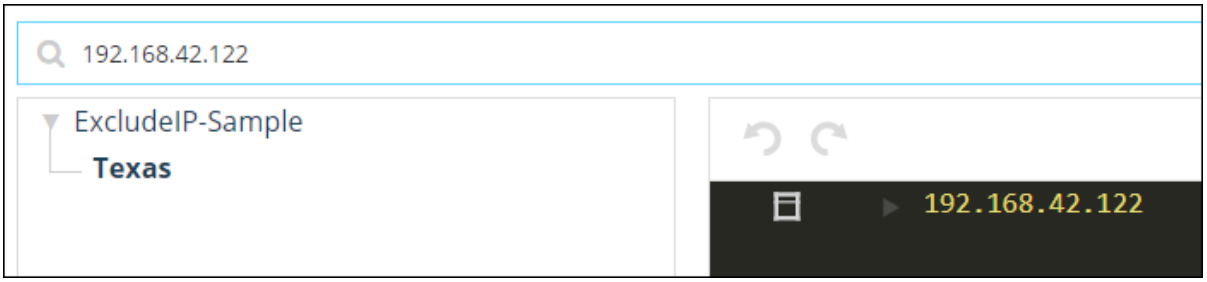


2. From the command bar on the top right of the screen, click **(Modify)** icon.
3. Update the data in the **Value** field.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. Click **Save**.
   The collection is modified with the number of documents in the collection now at 24. You can view the collection details and search for the IPs added to the collection.



Using Collections

Collections can be used in workflows either through APIs or to build a custom query (hooks).

In API

Sample Payload for adding documents to collections

```
{
  "payload": {
        "collection":"test"
        "action": "add",
        "documents": [
        {
        "deviceName": "F5",
        "ip": "1.2.3.4"
        },
        {
        "deviceName": "A10",
        "ip": "15.25.35.45"
        }
        ]
  }
}
```

In Query Explorer

Click here for more information on importing custom data into the Query Explorer.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Hooks Inventory

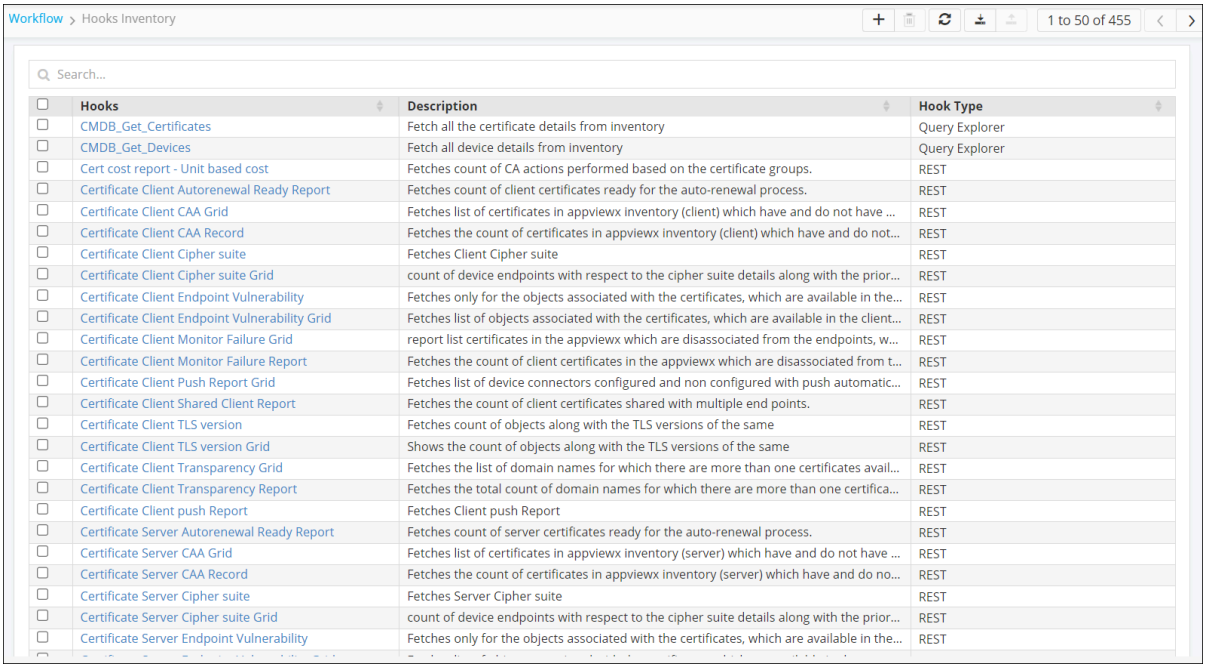
“Design and Define a Query once, reuse them forever…”

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.

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).

There are 3 ways of querying data through hooks:

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



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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Options available on the Hooks Inventory page (continued)

Option	Description
	Allows you to delete hook(s).
	Refreshes the hooks inventory.
	Allows you to import a hook.
	Allows you to export hook(s).

- Accessing the Hooks Inventory
- Creating a Query Explorer Hook
- Creating a Script Hook
- Creating a REST API Hook
- Defining a Hook and using it within a Workflow

Accessing the Hooks Inventory

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Hooks**. The **Workflow** > **Hooks Inventory** page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Hooks**. The **Workflow** > **Hooks Inventory** page is displayed.

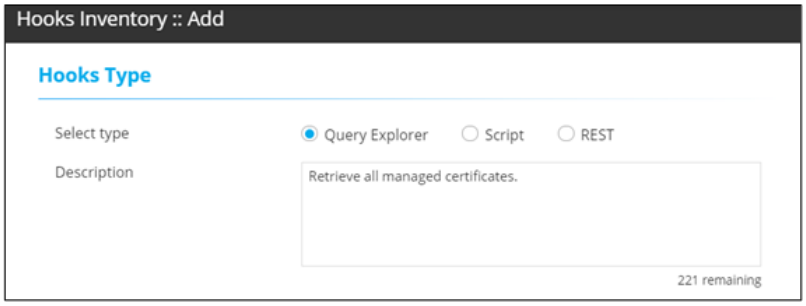
Note: For more information on how to switch between menus, click here.

Creating a Query Explorer Hook

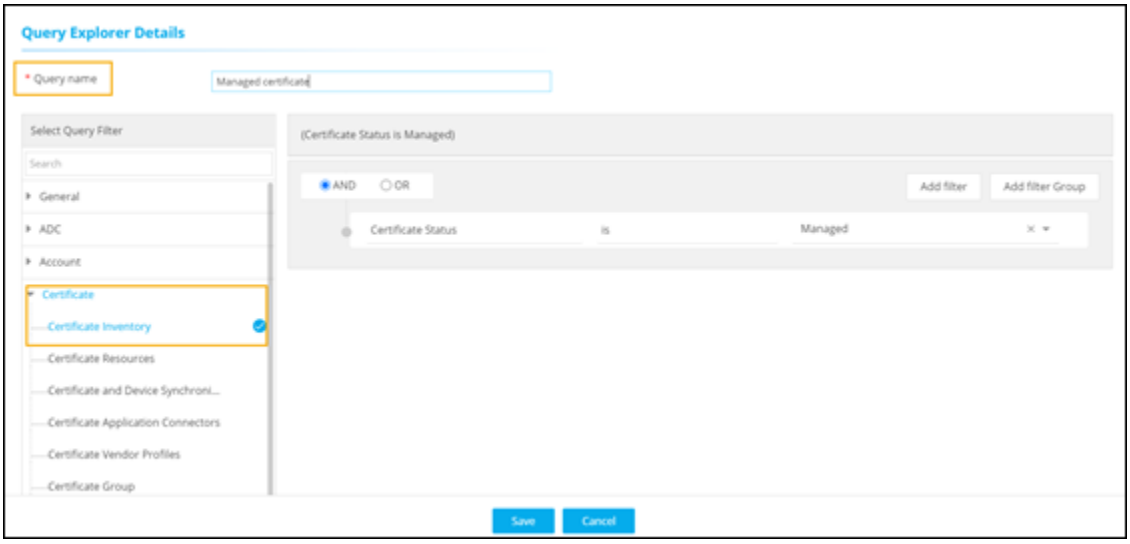
1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, from the command bar on the top right corner of the screen, click **(Add)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. On the **Workflow** > **Hooks Inventory** > **Add** page, under **Hooks Type**, select **Query Explorer**.
4. Enter a description for the hook.



5. Under **Query Explorer Details**, enter or select the required field information.



Field Descriptions for Query Explorer Details

Field	Description
***Query name**	Enter a valid query name. For example, **Managed certificates**.
Select Query Filter	Select an appropriate query filter from the pre-populated list of filters. For example, **Certificate Inventory under Certificate section**.
Add Filter	Add a filter and select appropriate fields from the predefined values. For example, **Certificate Status is Managed**.

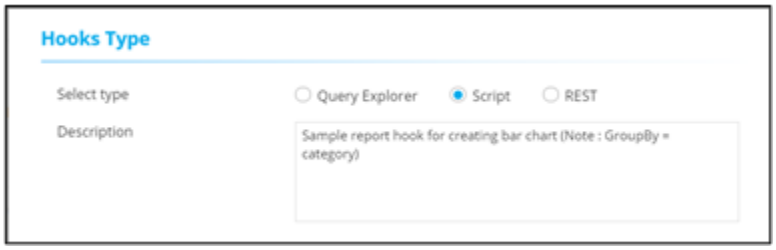
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
AND/OR	Select the appropriate conditional parameters depending on the type of output required in the report. **AND**: Report will reflect data satisfying all the filter conditions. **OR**: Report will reflect data satisfying either of the filter conditions.
** : Mandatory fields*	

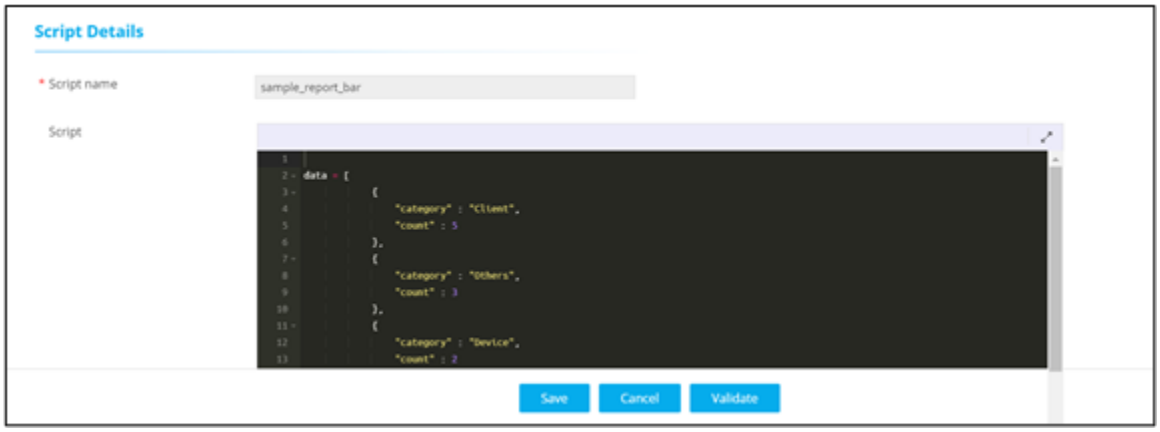
6. To add this new hook to the hooks inventory, click **Save**.

Creating a Script Hook

1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, from the command bar on the top right corner of the screen, click **(Add)** icon.
3. On the **Workflow** > **Hooks Inventory** > **Add** page, under **Hooks Type**, select **Script**.
4. Enter a description for the hook.



5. Under **Script Details**, enter the **Script name** and define the script logic.

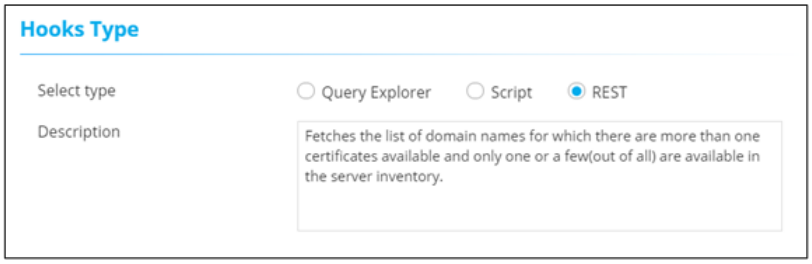


6. To add this new hook to the hooks inventory, click **Save**.

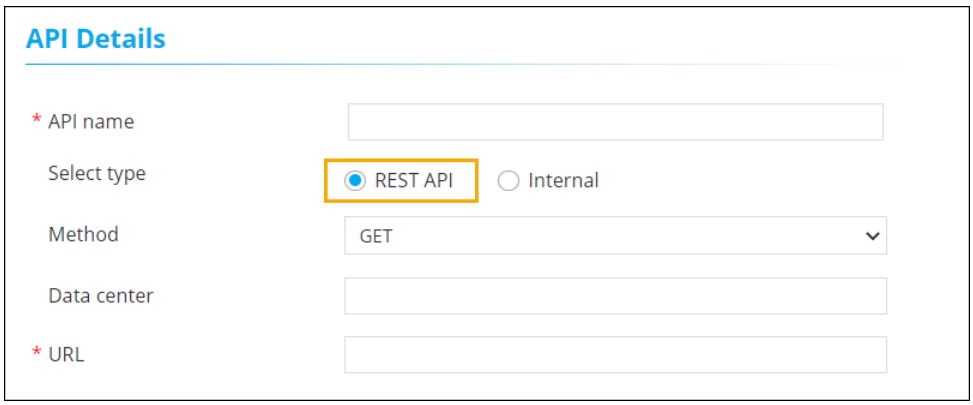
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Creating a REST API Hook

1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, from the command bar on the top right corner of the screen, click **(Add)** icon.
3. On the **Workflow** > **Hooks Inventory** > **Add** page, under **Hooks Type**, select **REST**.
4. Enter a description for the hook.



5. Under **API Details**, enter or select the required field information.
   - **Using REST API**: This is an external facing customer API (For example, ServiceNow, JIRA, GIT)



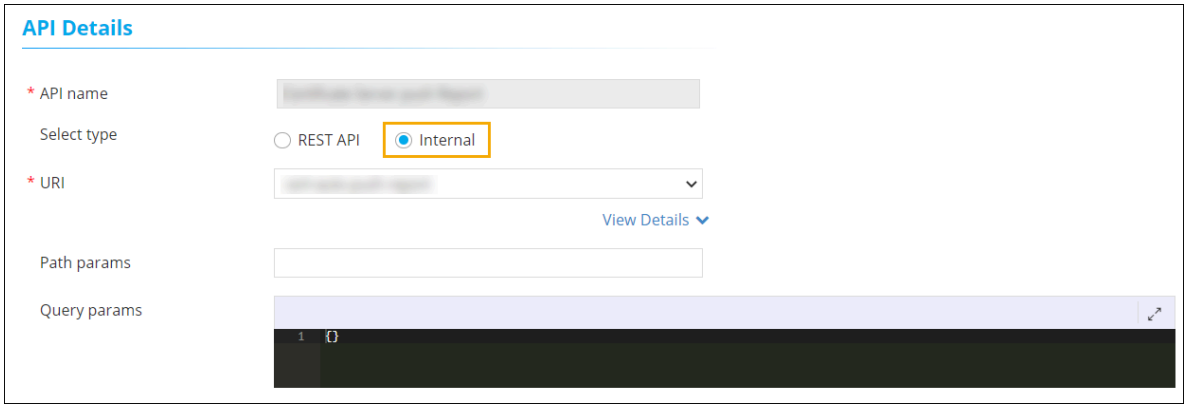
Field descriptions for API Details Select type REST API

Field	Description
API name	Enter a valid name for the API.
Select type	You can select between REST API and Internal. • **REST API:** Use this to select external API and enter the URL. • **Internal:** Use this to select an internal API and select a URI (Uniform Resources Identifier) from the dropdown menu.
Method	Select the API method for submitting requests.
***URL**	Enter the API URL.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Data center	Enter the name of the data center.
Use proxy	Select the checkbox for using a proxy and then select the **Host** from the dropdown. **Note:** Before enabling the proxy, add the proxy details under the Menu > Settings > General > Proxy.
** : Mandatory fields*	

◦ **Using Internal API**: API that is integrated internally



Field descriptions for API Details Select type Internal

Field	Description
API name	Enter a valid name for the API.
Select type	You can select between REST API and Internal. • **REST API**: Use this to select external API and enter the URL. • **Internal**: Use this to select an internal API and select a URI (Uniform Resources Identifier) from the dropdown menu.
Path params	Parameters that are used to identify a specific resource(s).
Query params	Parameters that are used to filter resources.
** : Mandatory fields*	

6. Under **Request Body**, enter the payload and response mapping details.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Request Body

Field	Description
Payload	Payload is the actual data that needs to be passed for the API.
Response mapping	Mapping an integration response from the backend.

7. To add this new REST API hook to the hooks inventory, click **Save**.

Using the REST API: An Example

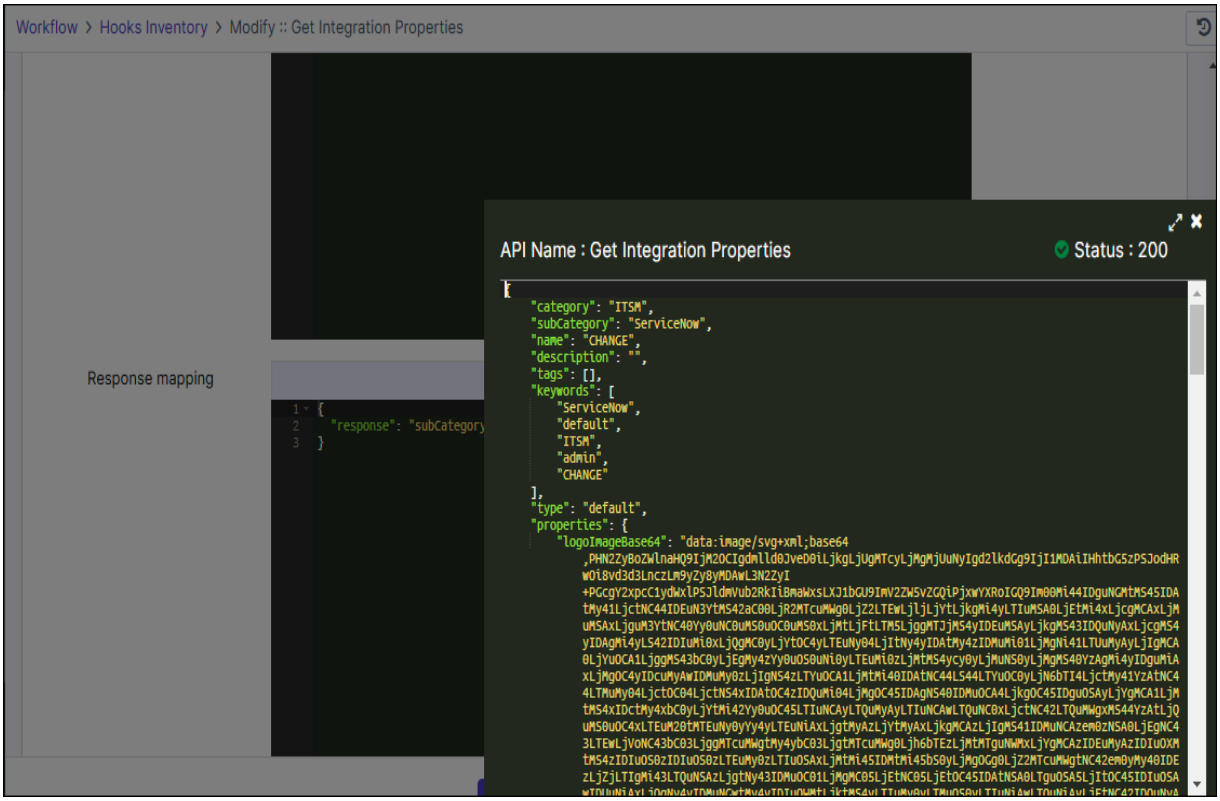
1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, search for the hook that was created. For example, *Get Integration Properties*.
3. Click on the *Get Integration Properties* to open the Modify - <Hook Name> page. (in this case the*Modify :: Get Integration Properties* page.
4. From the URI field, select the desired endpoint (For example: *visualworkflow-integration-get-subcategory*.
5. In the **Payload** field, provide the payload as follows:

```
{
  "category": "ITSM",
  "subCategory": "ServiceNow",
  "source": "",
  "encoded": false,
  "name": "CHANGE"
}
```

6. Click the **Validate** button.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The response for the selected API endpoint will be listed in a new window.



Using the Response Mapping in REST: An Example

This example demonstrates the use of the **Response Mapping** field in the REST API hook. Using the **Get Integration Properties** hook as an example, we will explore how Response Mapping is utilized within a Workflow.

1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, search for the hook that was created. For example, *Get Integration Properties*.
3. To view the REST API hook details, click *Get Integration Properties*.
   The *Modify::Get Integration Properties* page is displayed with the **Response Mapping** field containing the `"response": {}`
4. Based on the example in the previous section **Using the REST API: An Example**, from the API response that is fetched on clicking **Validate**. Note any of the **key** for which the response is expected in the workflow.
   Let us consider the key "subCategory" that should return the value `ServiceNow` when used in the workflow.
5. In the Response Mapping field, enter the key as follows:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"response": "subCategory"
```

6. Click **Save**.
7. Go to **Workflow** > **Magic Variables**.

   The **Magic Variables** window is displayed.
8. Click **+** (Add) button, and enter the details as follows:
   - *__Variable name__= *mytest* [Name for the magic variable]
   - **Type** = *Dynamic*
   - *__Key__ = *test_hook* [The key must not contain any spaces in the value entered]
   - *__Hooks__ = *Get Integration Properties* [Hook created in the above examples]
9. Click **Add**.
10. Go to **Workflow** > **Design** > **Create New** (button).

    The workflow design page is displayed.
11. Enter the name for the new workflow in the **Name** field (*hook_api_test*) and click **Save**.

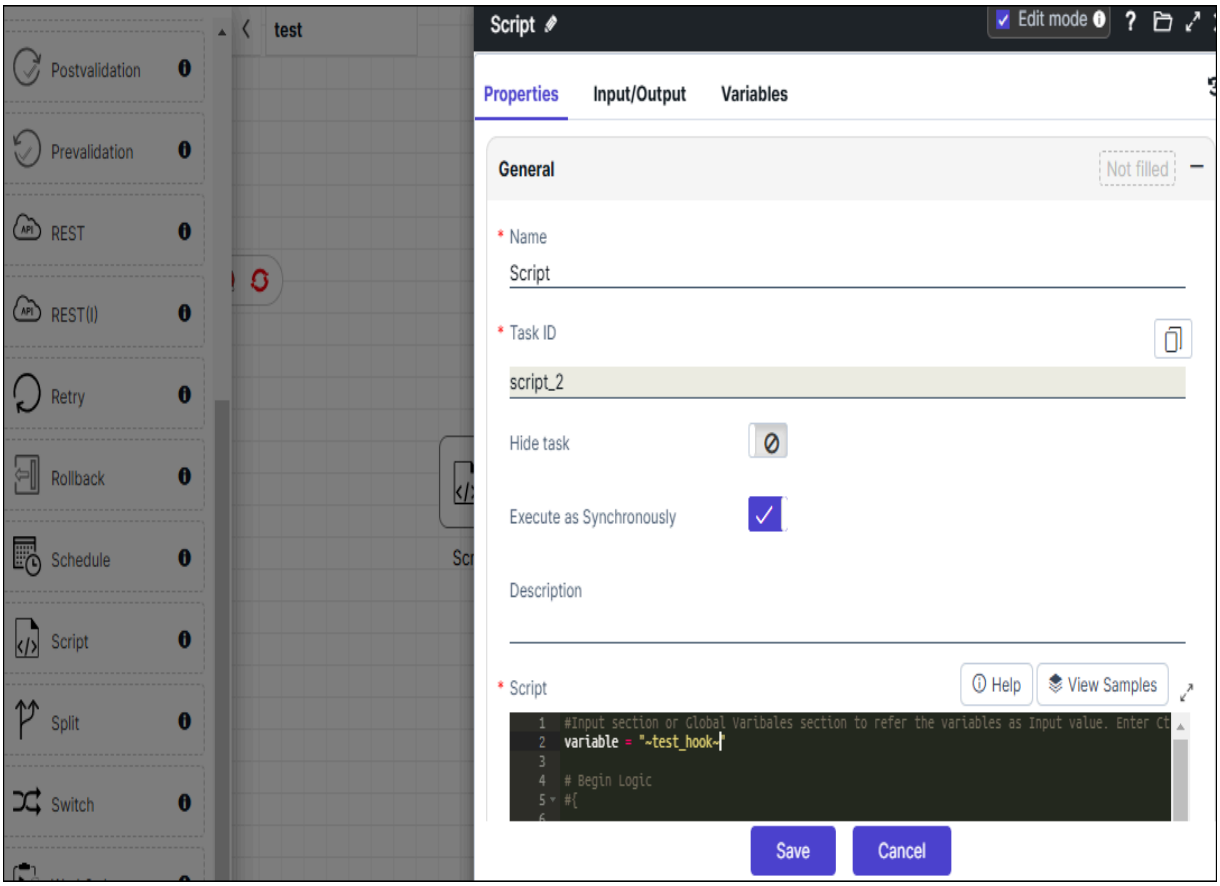
    The new workflow design page is displayed with **Start** and **Stop** buttons.
12. From the left panel, click **General** and then drag nd drop the **Script** task to the workflow design page.

    The **Script** edit window is displayed with the default tab **Properties**.
13. In the **Script** editor section, enter the Magic Variable Key, *test_hook* in the variable field as follows:

```
variable = "~test_hook~"
```

Note: The key should begin and end with the tilde (~) symbol.

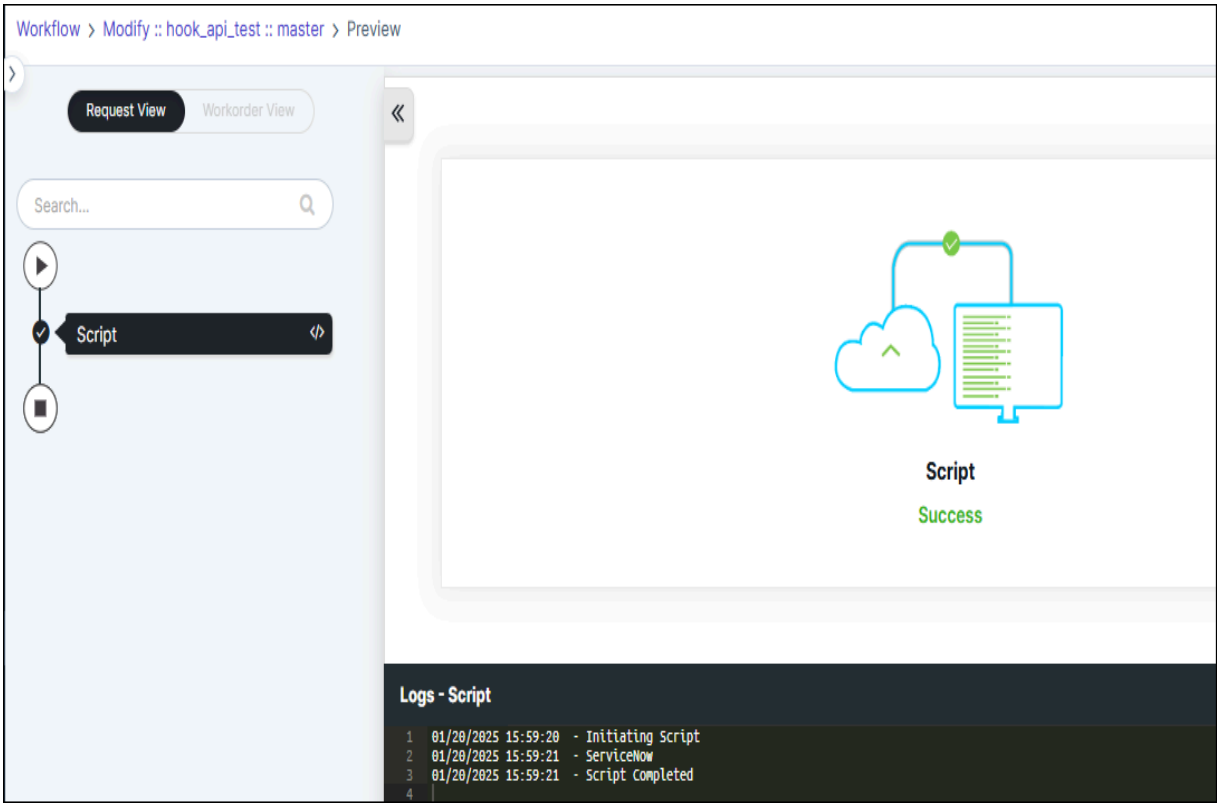
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



14. Click **Save**.
15. Execute the workflow by clicking the **Preview Classic** button.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The workflow should execute successfully and the logs window will display the subCategory `ServiceNow`.

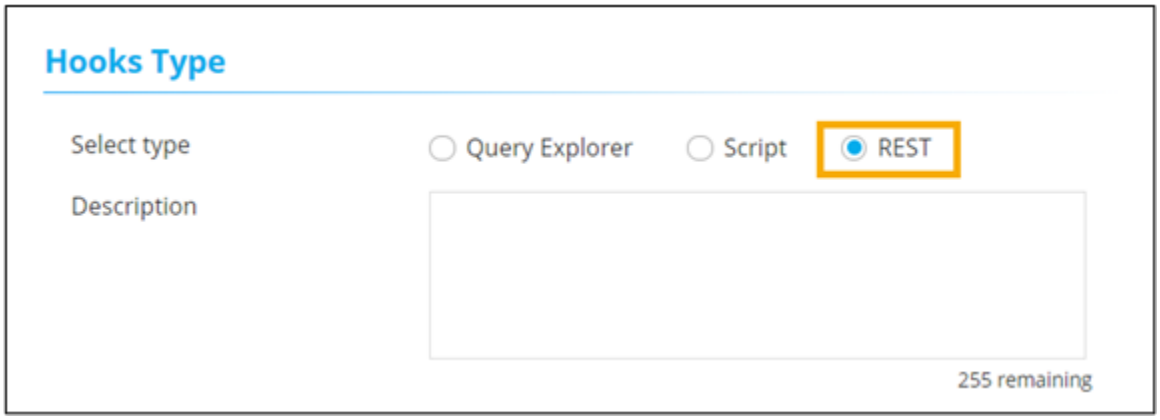


Defining a Hook and using it within a Workflow

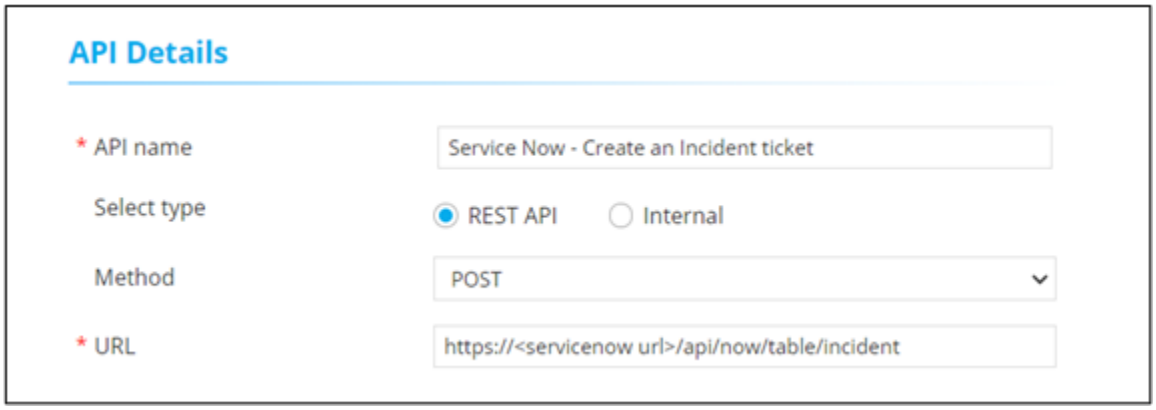
To define a common hook to create an incident ticket on ServiceNow and use it within a workflow:

1. Go to the Hooks Inventory.
2. On the **Workflow** > **Hooks Inventory** page, from the command bar on the top right corner of the screen, click **(Add)** icon.
3. On the **Workflow** > **Hooks Inventory** > **Add** page, under **Hooks Type** select **REST** and enter a **Description** for the hook.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. In the **API Details** section, enter or select the required field information as shown in the image below.

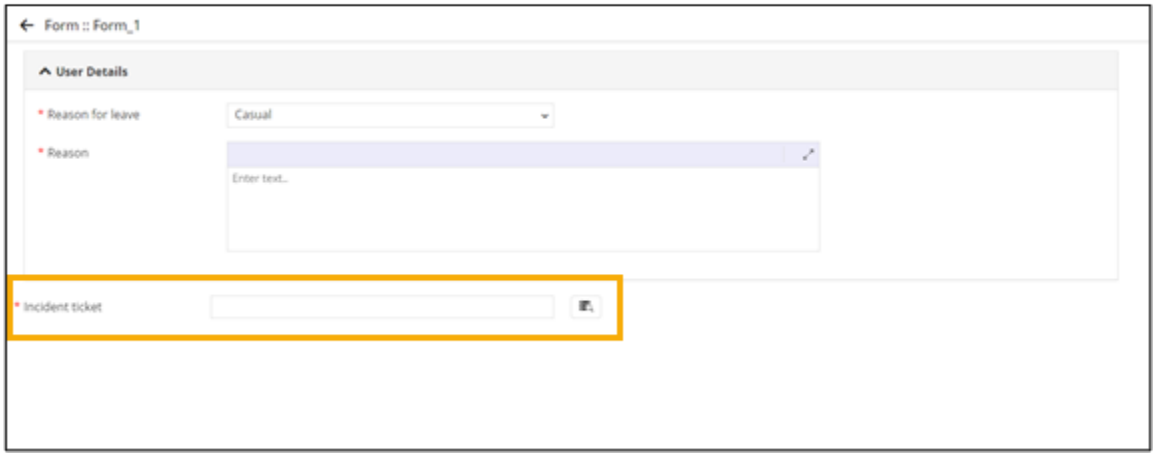


5. Click **Save**.
6. Design a workflow.
7. From the **User Interface** section, drag and drop a **Form** task.
8. Under the **Form builder** tab, define the necessary form fields.
9. Under the **Hooks** tab, select the **Hook type** as **REST**.
10. From the **Hooks Inventory** dropdown, select the API name as defined before.
11. From the **Field ID** dropdown, select the field against which the hook should be mapped.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

12. Click **Add**.
13. Under the **Form Builder** tab, click **(View)** icon.

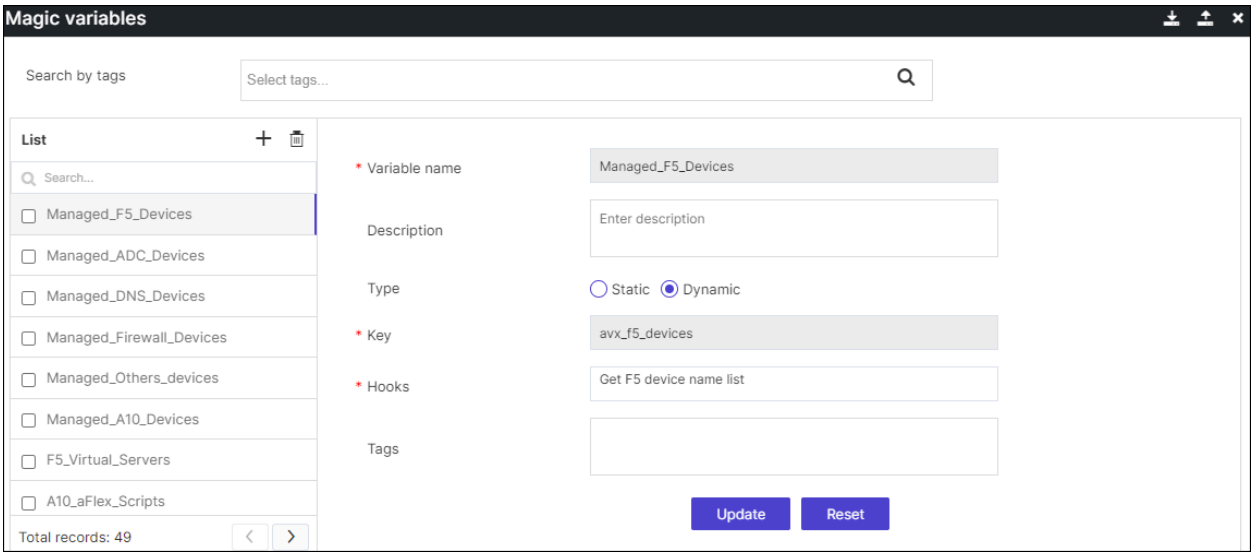
    A preview of the form shows the hook added to the form field.



Magic Variable

This shows a list of all the Magic variables defined in various tasks across workflows.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

Accessing Magic Variables

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Magic Variables**. The **Magic Variables** pop-up window is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Magic Variables**. The **Magic Variables** pop-up window is displayed.

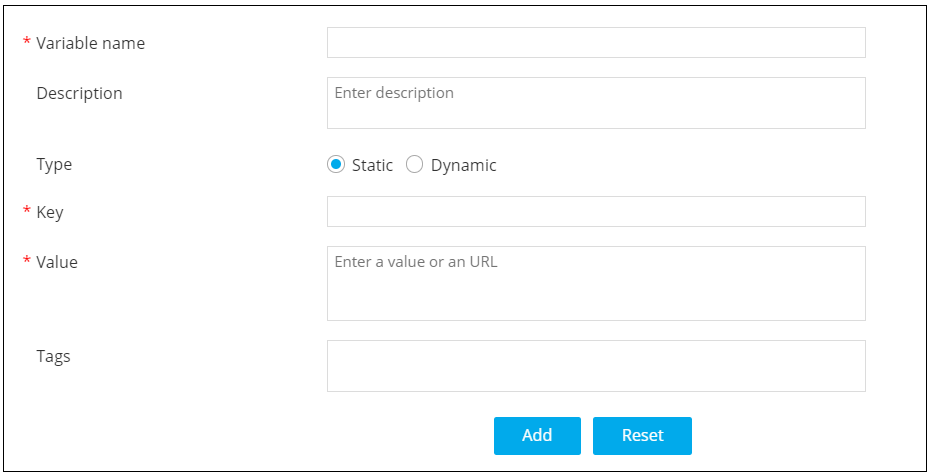
Note: For more information on how to switch between menus, click here.

Defining a Hook and declaring it as a Magic Variable

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. From the Hooks inventory, create a Hook using REST API or Script.
2. Go to the Magic Variables section.
3. In the **Magic variables** window, to define a new magic variable, click + **(Add)** icon.
4. Enter or select the required field information.



Field descriptions for adding new magic variable

Field	Description
***Variable name**	Enter the name for this magic variable. For example: **Managed_F5_Devices**
Description	Enter a description for the magic variable.
Type	Select if the magic variable is • **Static:** Hooks which are static in nature can be referred as magic variables. For example, Slack webhook, ITSM change ticket URL etc. • **Dynamic:** Hooks that execute dynamically can be referred 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.
***Key**	Enter a unique key for the magic variable.
***Value**	Enter the values for the magic variable. These can either be comma-separated values or a URL. **Note:** This field is displayed only when you select the **Type** as **Static**.
***Hooks**	Enter the name of the hook that you want to define as the magic variable. For example, **Get F5 device name list**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

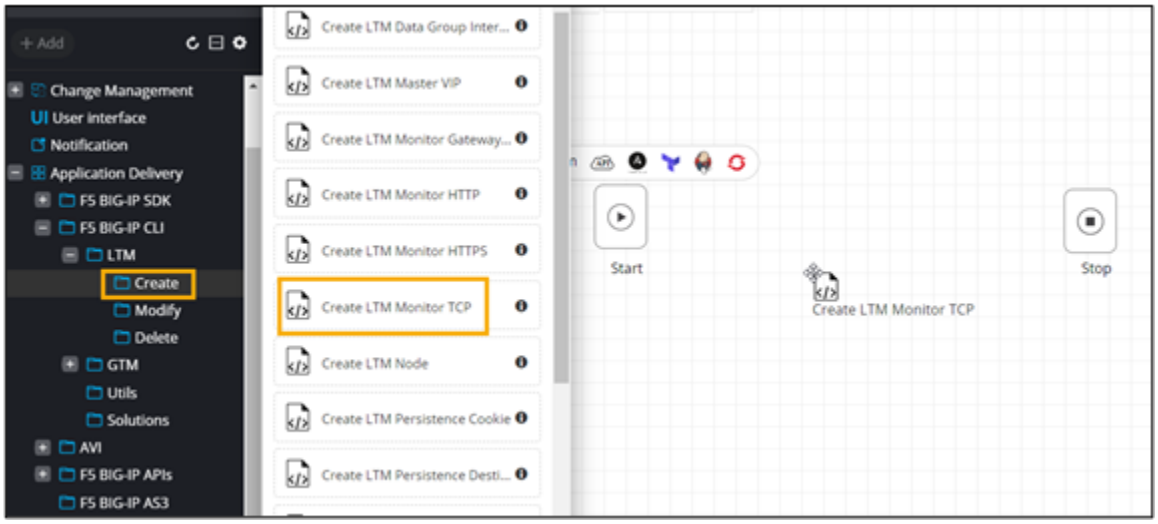
Field	Description
	Note: This field is displayed only when you select the **Type** as **Dynamic**.
Tags	Enter tags to be associated with this magic variable, if any. This will help you to search for magic variables by tags.
** : Mandatory fields*	

5. Click **Add**.

Using a Magic Variable within a Workflow

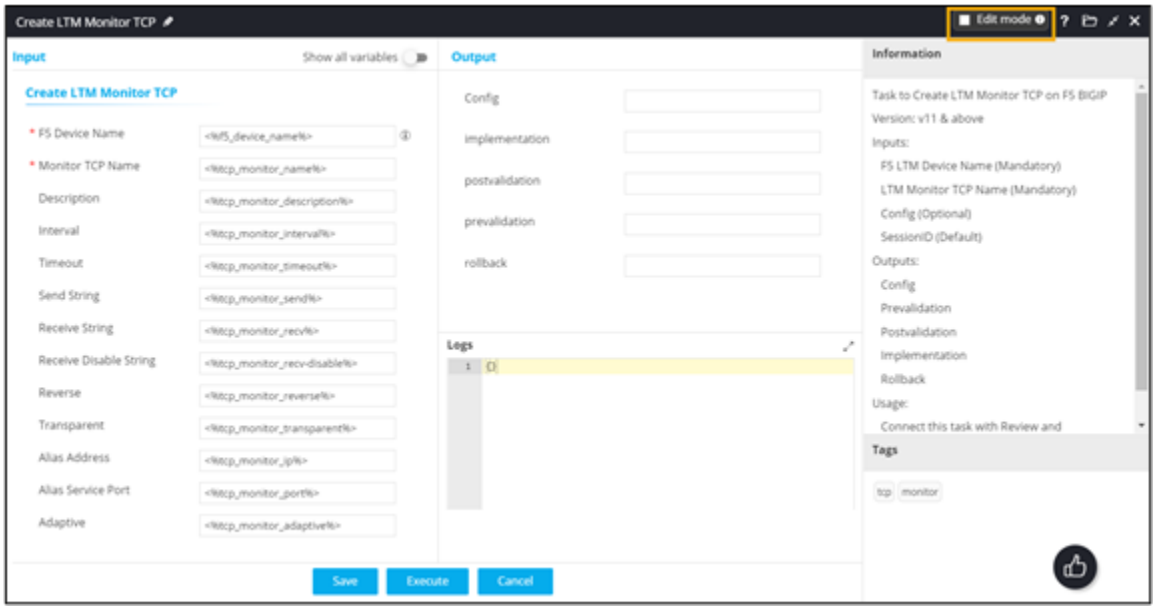
This section describes the steps to define a hook as a magic variable and use it in the Form task.

1. Design a workflow.
2. In the design workspace, 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.



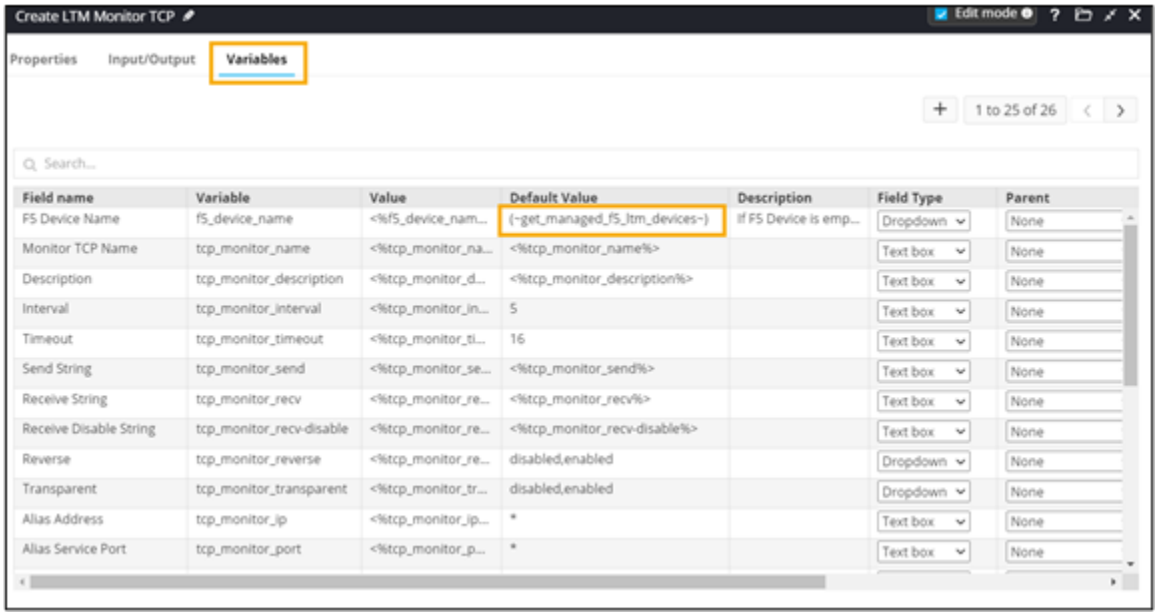
4. In the **Create LTM Monitor TCP** task window, select the **Edit mode** checkbox to see information on the variables used in this task.

Copyright © 2025 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**.

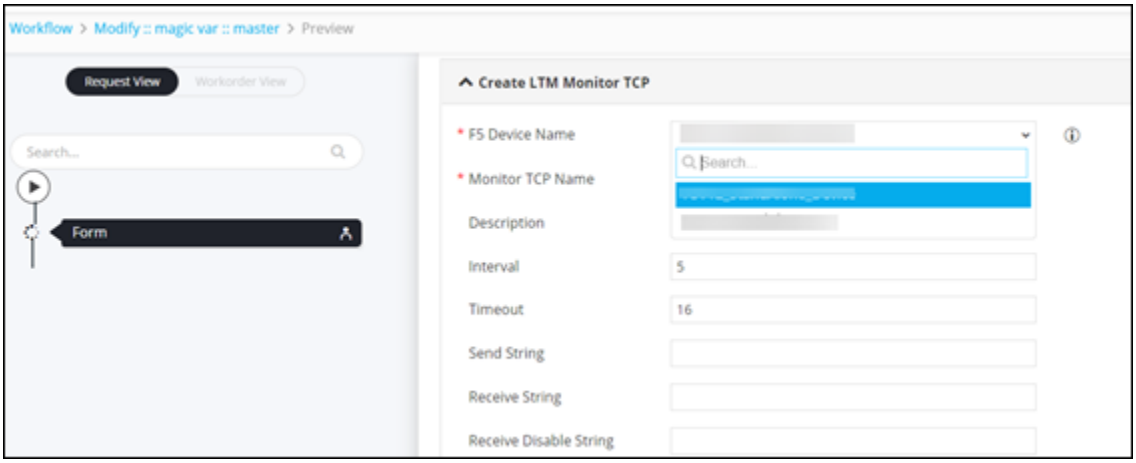
   The magic variable `(~get_managed_f5_ltm_devices~)` retrieves the list of all managed F5 LTM devices.



6. Auto-generate a self-service form for this workflow.
7. Connect the workflow tasks and click **Preview**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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



Helper Scripts

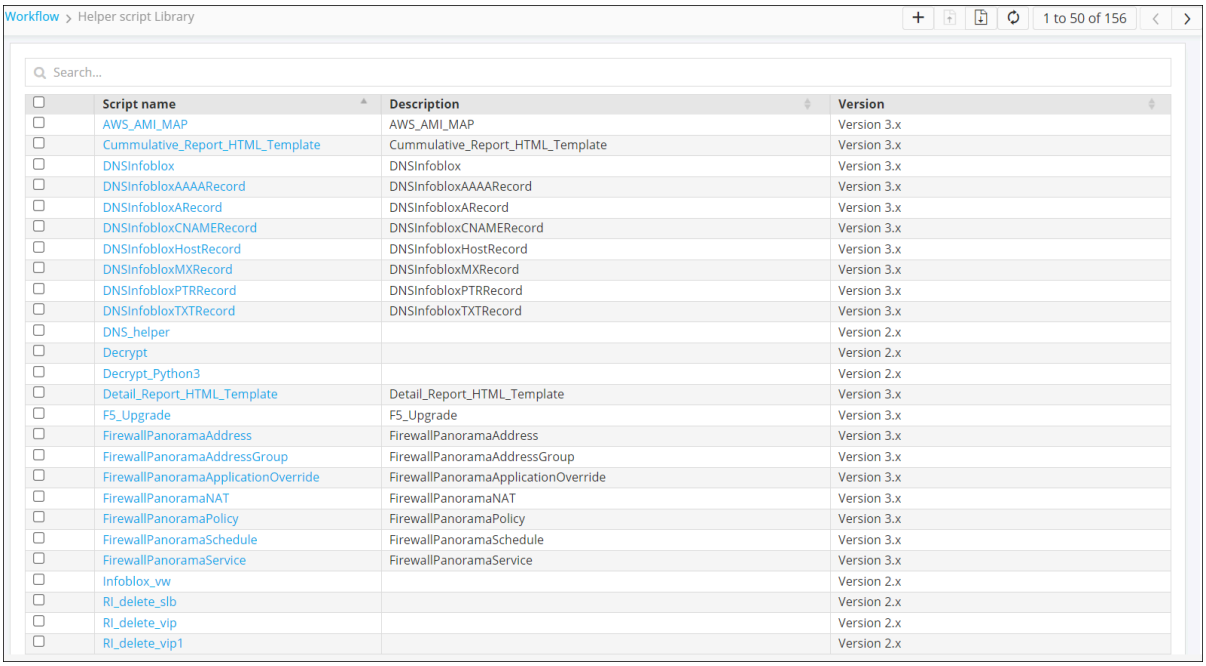
“Write once, reuse them forever…”

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.

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 **Helper Script Library** can be accessed by navigating to **Automation** > **WORKFLOW** > **Helper script**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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.

- Accessing Helper Script Library
- Creating a Helper Script
- Exporting Helper Scripts
- Importing Helper Scripts
- Configuring RBAC for Helper Scripts
- Using a Default Helper Script

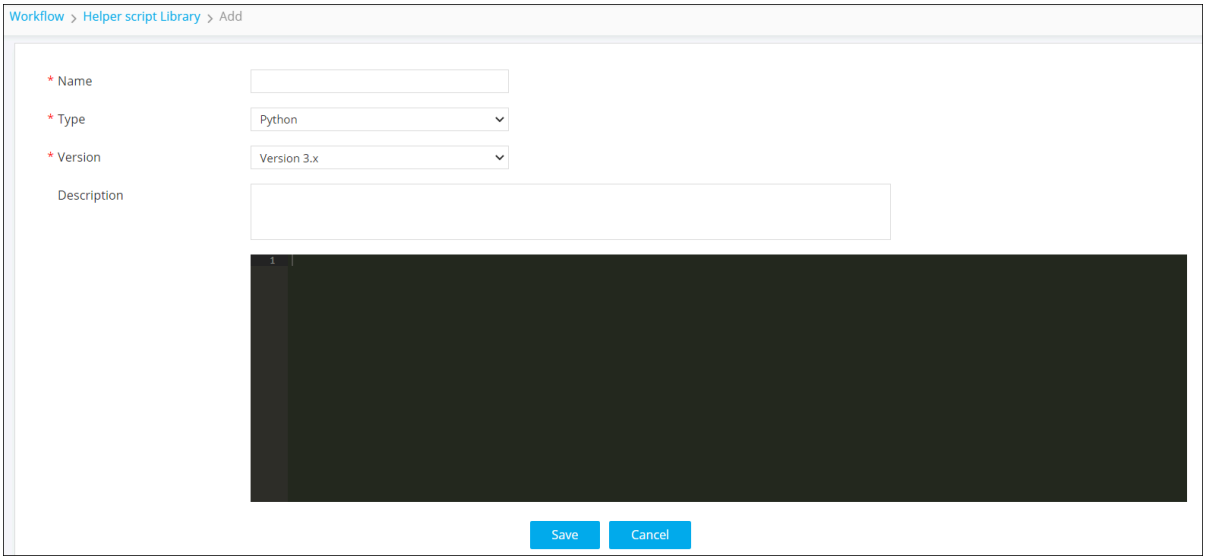
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Accessing Helper Script Library

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Helper Script**. The **Workflow** > **Helper script Library**page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Helper Script**. The **Workflow** > **Helper script Library**page is displayed.
Note: For more information on how to switch between menus, click here.	

Creating a Helper Script

1. Go to the Helper Script Library.
2. To create a new Helper Script, on the **Helper Script Library** page, click **(Add)** icon in the command bar.
3. On the **Workflow** > **Helper Script Library** > **Add** page, enter or select the field information..

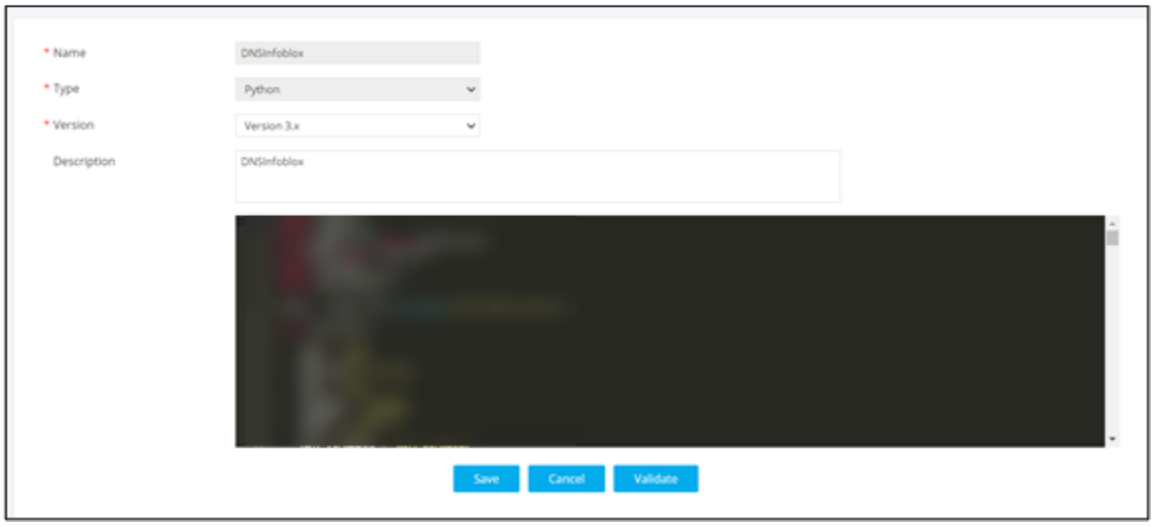


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for adding new workflow to the helper script library

Field	Description
***Name**	Provide an appropriate name for the Helper Script.
***Type**	Select script as Python.
***Version**	Select version as 3.x
Description	Enter a description of the script.
** : Mandatory fields*	

4. Define the script logic.

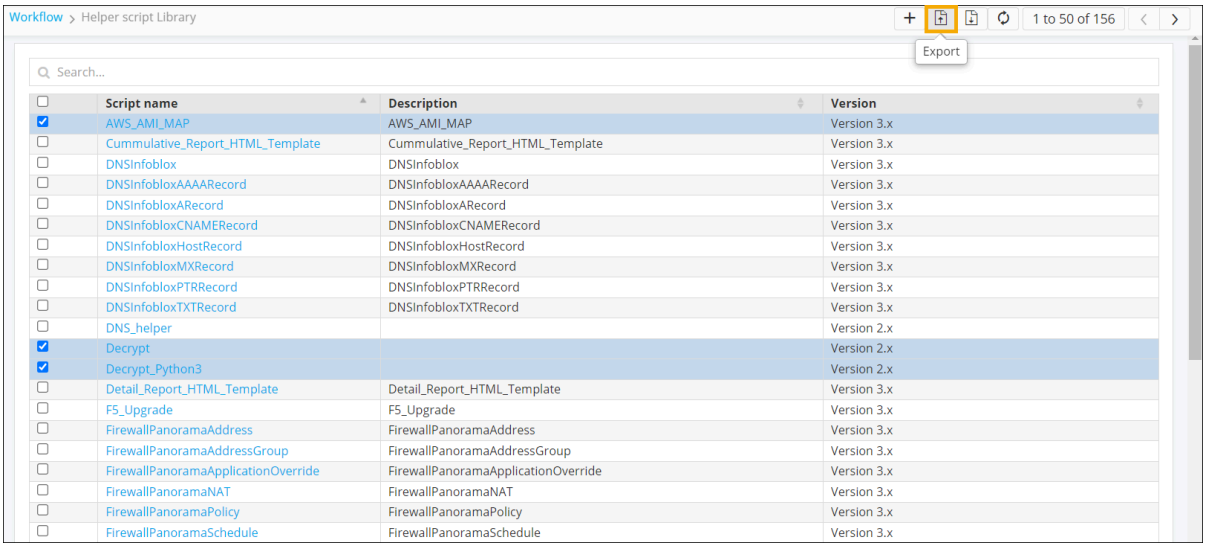


5. Click **Save**.

Exporting Helper Scripts

1. Go to the Helper Script Library.
2. On the **Helper Script Library** page, select the helper script(s) to be exported.
3. To export the selected helper script(s), click **(Export)** icon in the command bar.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

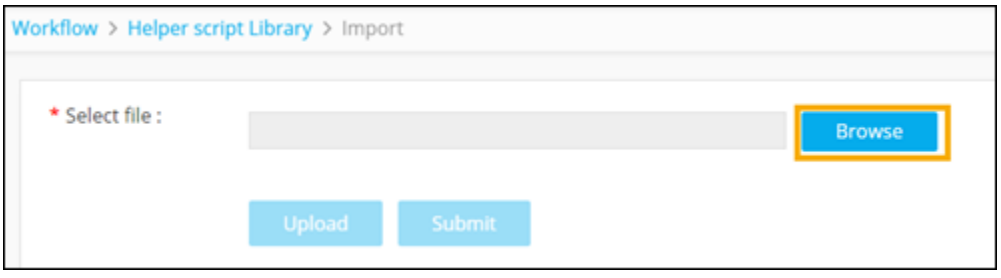


4. Click **Ok** in the **Confirmation** pop-up window.
   The helper script(s) is/are saved to your computer as a <.zip> file.

> **Note:** When you open the exported file in a text editor, you will notice \n appended in many places within the script. This is the default behavior. However, when the same file is imported, these characters will not appear in the script.

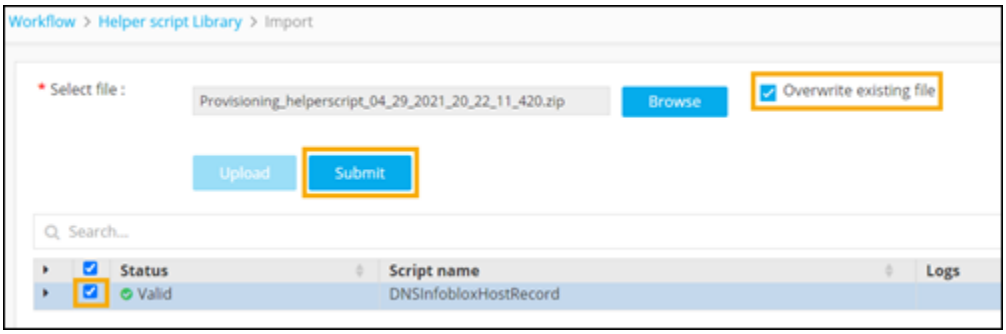
Importing Helper Scripts

1. Go to the Helper Script Library.
2. On the **Helper Scripts Library** page, click **(Import)** icon in the command bar.
3. To select the helper script file to be uploaded, click **Browse**.



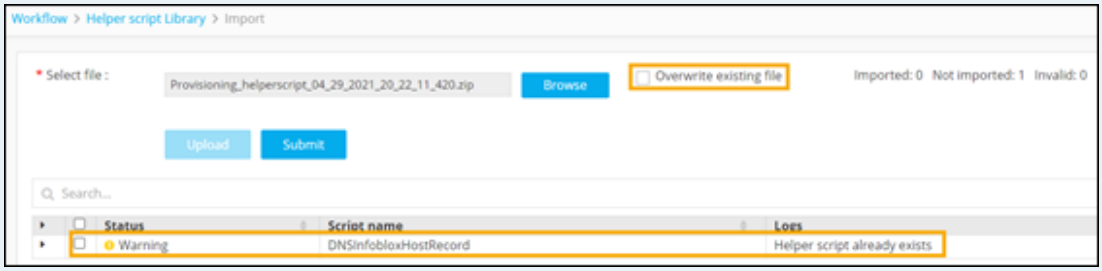
4. To upload the selected file, click **Upload**.
5. Select the script and click **Submit**.

Copyright © 2025 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 warning will be displayed to the user on clicking Submit.



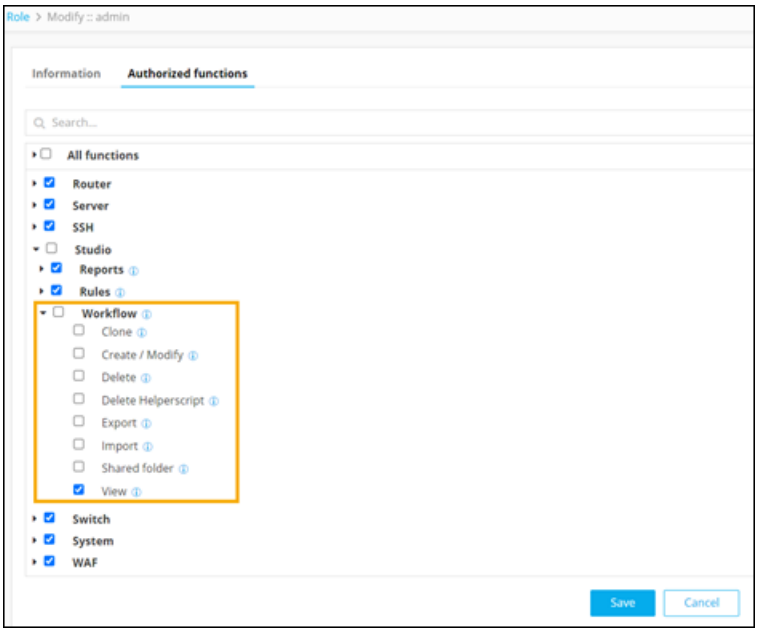
Configuring RBAC for Helper Scripts

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.

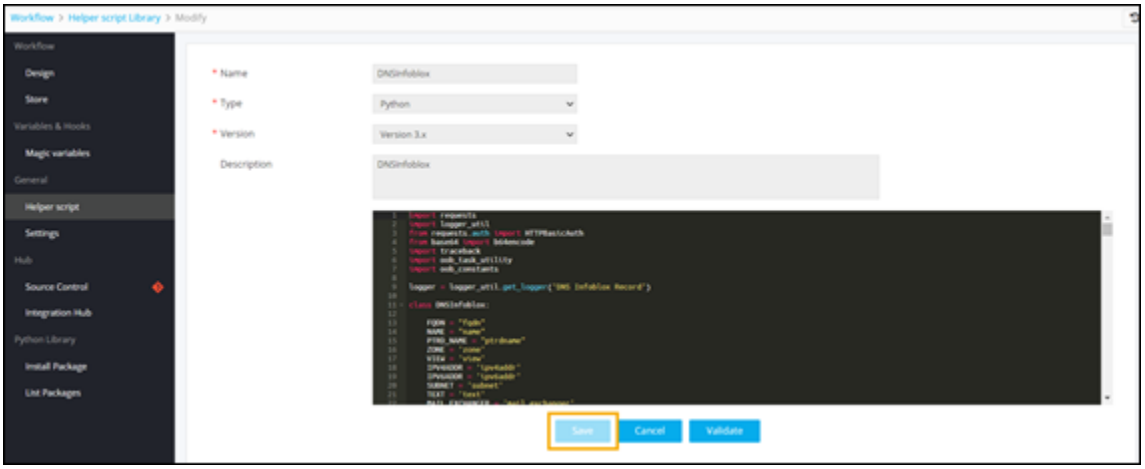
To restrict a user from modifying Helper Scripts in the Workflow Studio:

1. From the upper left corner of the screen, click **(Menu)** icon.
2. From the menu displayed, select **Platform** > **IDENTITY** > **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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

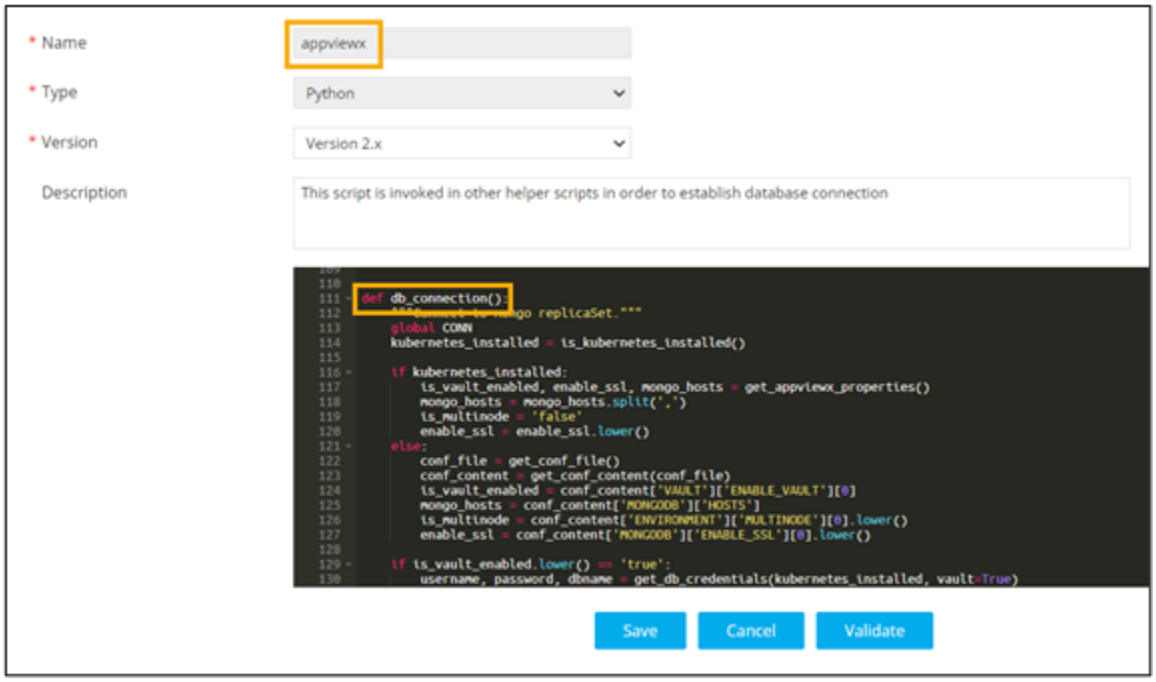


Using a Default Helper Script

1. Leverage the helper script called "appviewx", which establishes database connection.

   Function name: `db_connection`

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

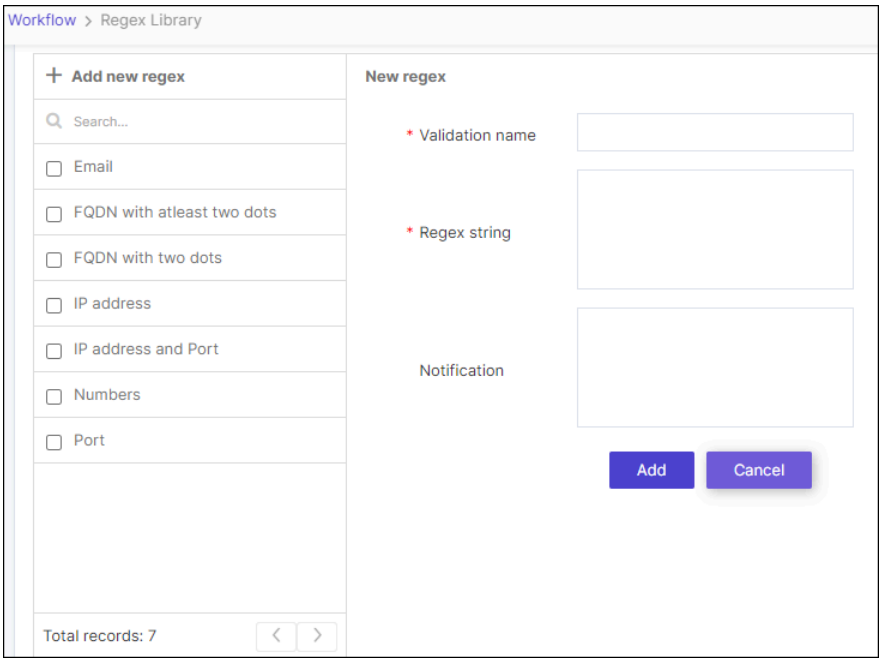
Get device associate script

```
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 © 2025 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.



- Accessing the Regex Library
- Adding a New Regex
- Modifying an Existing Regex

Accessing the Regex Library

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Regex Library**. The **Workflow** > **Regex Library**page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Regex Library**.

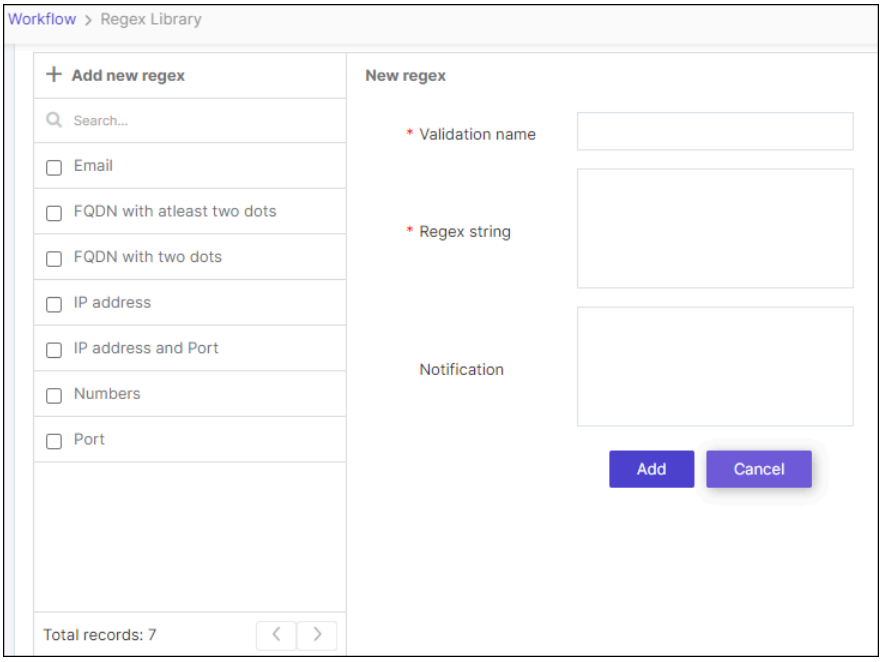
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

New Menu	Old Menu
	The **Workflow** > **Regex Library**page is displayed.

Note: For more information on how to switch between menus, click here.

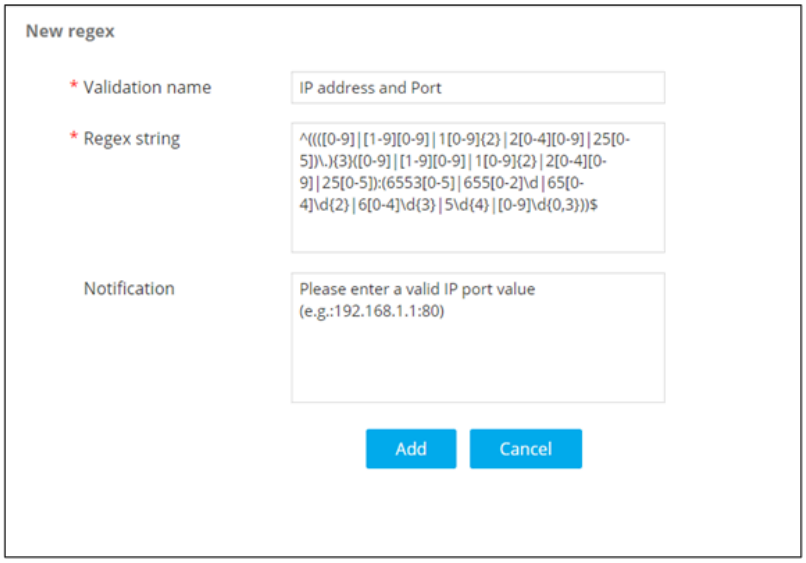
Adding a New Regex

1. Go to the Regex Library.
2. To add a new regex string, on the **Workflow** > **Regex Library** page, click **Add new regex**.



3. Enter the required field information as shown.

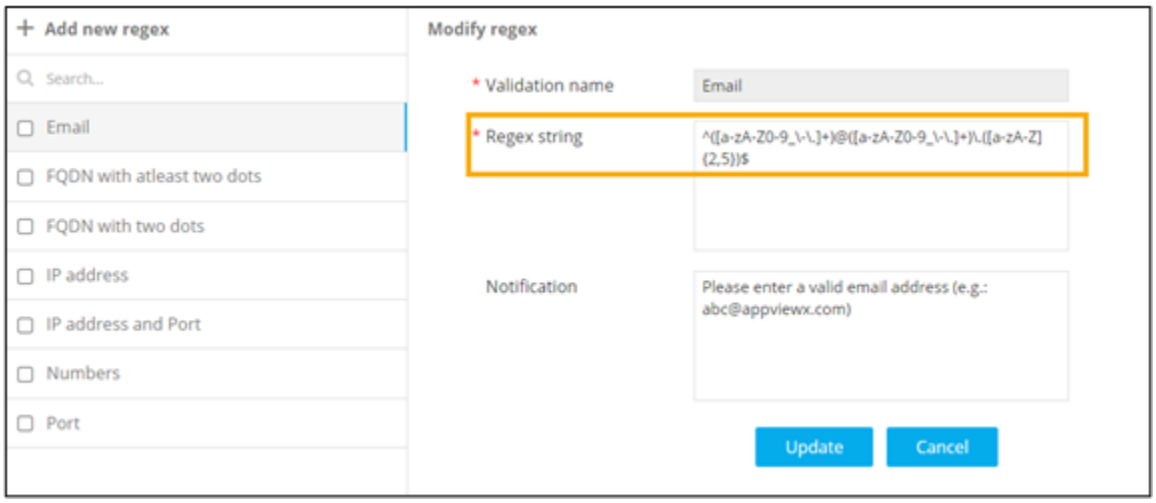
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

Modifying an Existing Regex

1. Go to the Regex Library.
2. On the **Workflow** > **Regex Library** page, select an existing Regex from the list. For example: **Email**.
3. To modify the **Regex string**, enter a different regex pattern for email validation.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Settings

You can request specific customized request settings, allowing tailored configurations for efficient and personalized automation workflows.

To define the request settings:

New Menu	Old Menu
1. Form the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, click **WORKFLOW** > **Request Settings**.	1. Form the main navigation menu, select **Settings**. 2. On the **Design** page, **Request Settings**popup is displayed.
Note: For more information on how to switch between menus, click here.	

1. When you click **Request Settings** the **Settings** popup page is displayed.
2. Under the **Request Settings** section, configure the request settings based on request execution access, assigned request filters, and role-based access.
3. In the **Request Execution Access** field, select the execution access to **All User** or **Requestor** or **Dynamic Auth**.
   - If execution access is set to **All User**, every user with all assigned resources can view the request.
   - If execution access is set to **Requestor**, all users with submitter resource can view the request.
   - If execution access is set to **Dynamic Auth**, users with dynamic authentication can only view the request.
4. In the **Filter Assigned Request** field, select the filter assigned to **All User** or **Requestor** or **Dynamic Auth**.
   - If filter assigned is set to **All User**, every user with all assigned resources can view the request.
   - If filter assigned is set to **Requestor**, all users with submitter resource can view the request.
   - If filter assigned is set to **Dynamic Auth**, users with dynamic authentication can only view the request.
5. Turn on the **Role Based Access** toggle to view role based access to the request.
6. Click **Save**.

   The **Request Settings** will be updated successfully, and only the customized requests will be displayed on the Workflow page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Provisioning for Workflow Tasks

You can define specific configurations for executing the provisioning workflow tasks such as Implementation, Prevalidation, Postvalidation, and Rollback.

To define the configurations:

New Menu	Old Menu
1. Form the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, click **WORKFLOW** > **Provisioning**.	1. Form the main navigation menu, select **Settings**. 2. On the **Settings** module, from the left menu, click **Provisioning**.
Note: For more information on how to switch between menus, click here.	

1. Under the **Work order configuration** section, define the configurations for Both or Manual or Auto of the **Implementation type**.
2. In the **Implementation type** field, select the Implementation type from the dropdown.
3. Under the **Command configuration** section, define the configurations for success or failure of the implementation task.
4. In the **Failure messages** field, enter the keywords for error responses to identify the task as failed as shown in the image below.

   If the workflow execution displays any of the keywords defined above as error messages, the task will be considered as failed.

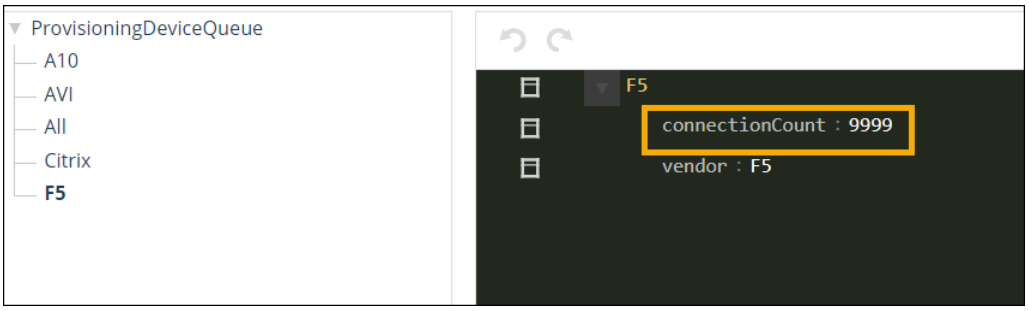
5. In the **Excluded failure messages** field, enter the keywords to be excluded for identifying the task as failed.
6. In the **Commands to wait** field, you can enter a set of commands and define a time out for the implementation task. If the task contains the configured commands, then it will time out as per the time out defined. This is useful for tasks or commands that take longer than the default time to execute.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. In the **Time out (mins)** field, enter the timeout value for the tasks having the configured set of commands.

> **Note:** The default value is 0.5 minutes or 30 seconds.

8. Turn on the **Continue on a command failure** toggle, to proceed in case of a command failure.
9. Turn on the **Enable Device Queue** toggle to enable queuing of device connections as per the defined connection count in the Collection.



10. Turn on the **Enable Device Session Cache** toggle, to check for any existing open sessions for the device and execute other requests in the same session.
11. Turn on the **Rollback work order** toggle to rollback configurations that are partially implemented.

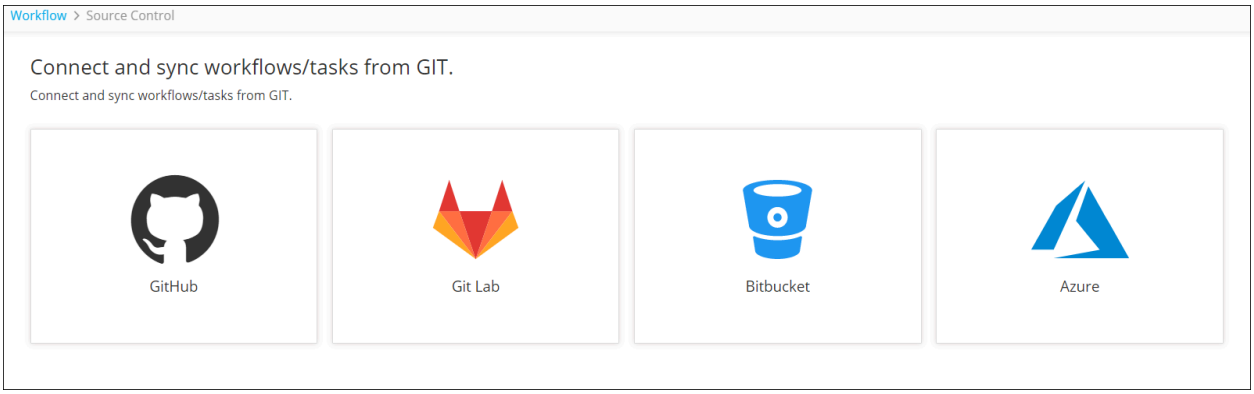
> **Note:** Configurations that were successfully implemented will be excluded from the rollback work order.

12. Under the **Rest configuration** section, define the different success status codes in the **Rest success status code** field.
13. Under the **Collection configuration** section, turn on the **Enable default collection** toggle to create a collection in the name of the device and store all the IPs in the device (using a predefined parsing logic) during the next device parsing.

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 © 2025 AppViewX, Inc. All Rights Reserved.



Note: Source control integration is not supported for SaaS.

- Prerequisites
- Accessing the Source Control Page
- Linking a Repository
- Committing Workflows from the Workflow Page
- Committing Workflows from the Design Canvas
- Committing Tasks and Subflows
- Pushing to a Remote Repository from Workflow Page
- Pushing to a Remote Repository from the Design Canvas
- Pulling from the Repository from Workflow Page
- Pulling from the repository from the Design Canvas
- 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Accessing the Source Control Page

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Source Control**. The **Workflow** > **Source Control** page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Source Control**. The **Workflow** > **Source Control** page is displayed.
Note: For more information on how to switch between menus, click here.	

Linking a Repository

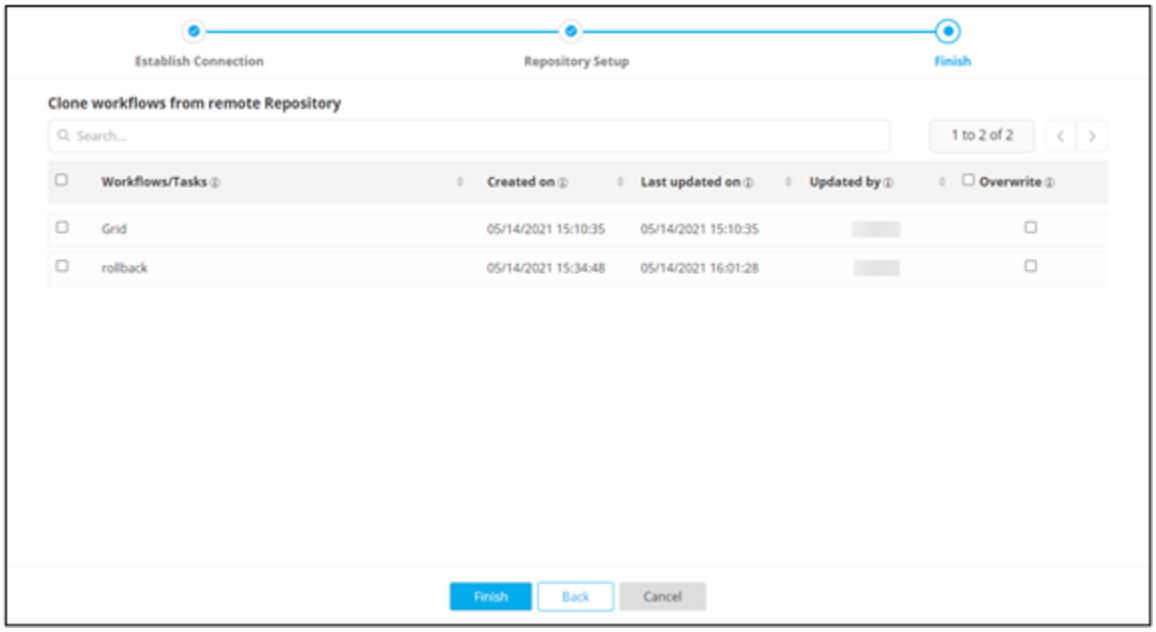
1. Go to the Source Control page.
2. On the **Source Control** page, select the integration type as **GitLab**.
3. Select the connection type as HTTPS or SSH.
4. For **HTTPS**, enter the valid credentials along with the HTTPS clone **URL**.
5. For **SSH**, enter the valid private key corresponding to the public key and SSH clone URL.

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

6. From the **Branches** dropdown, select **master**.
7. Click **Next**.

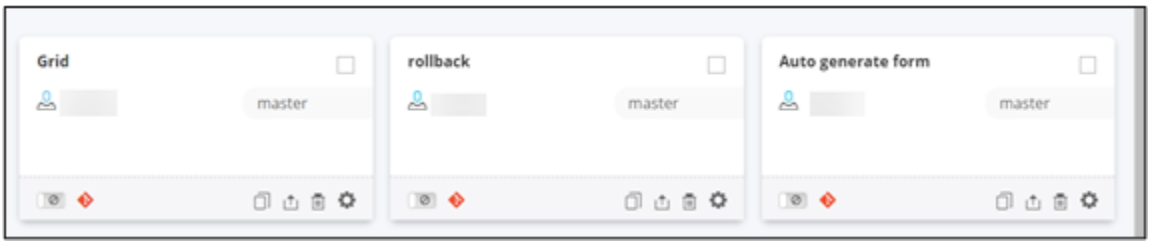
   All available workflows, tasks and subflows in the remote repository are displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

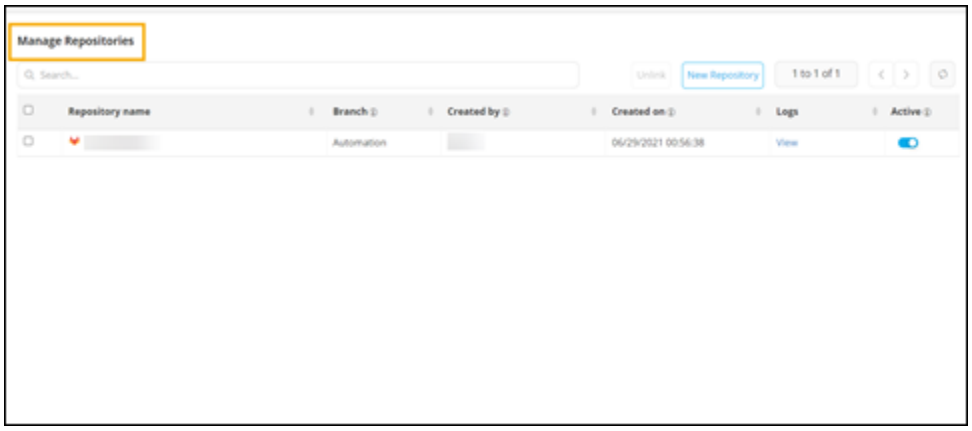


8. Select the workflows to be cloned and click **Finish**.
9. In the **Confirmation** pop-up window, click **Yes**.
   - Cloned tasks and subflows are added to Source Control Directory and Workflows to the inventory.

     The connected workflows reflect the Source Control logo .



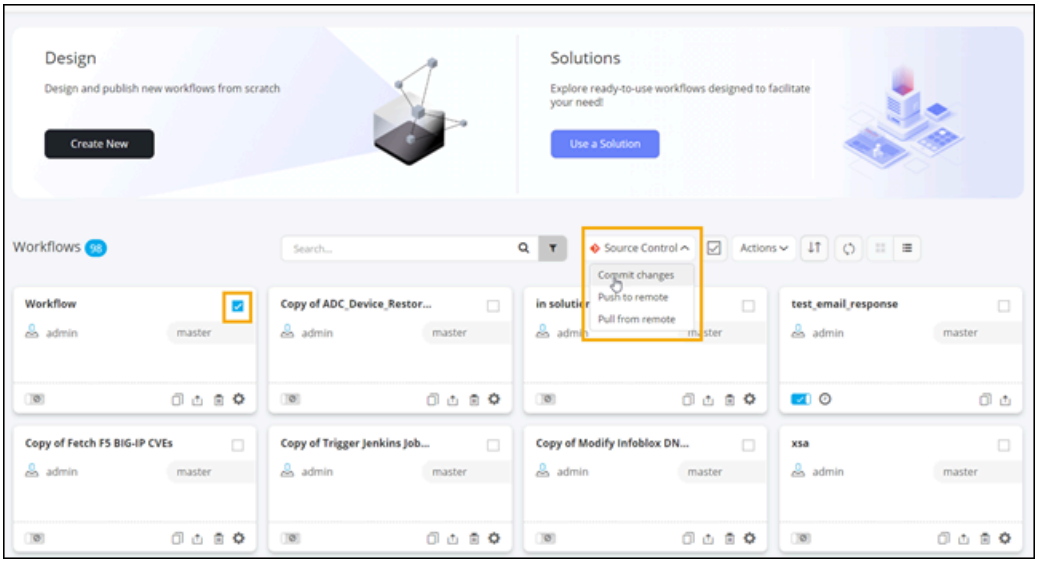
   - Cloned Repository is added to the **Manage Repositories** page.



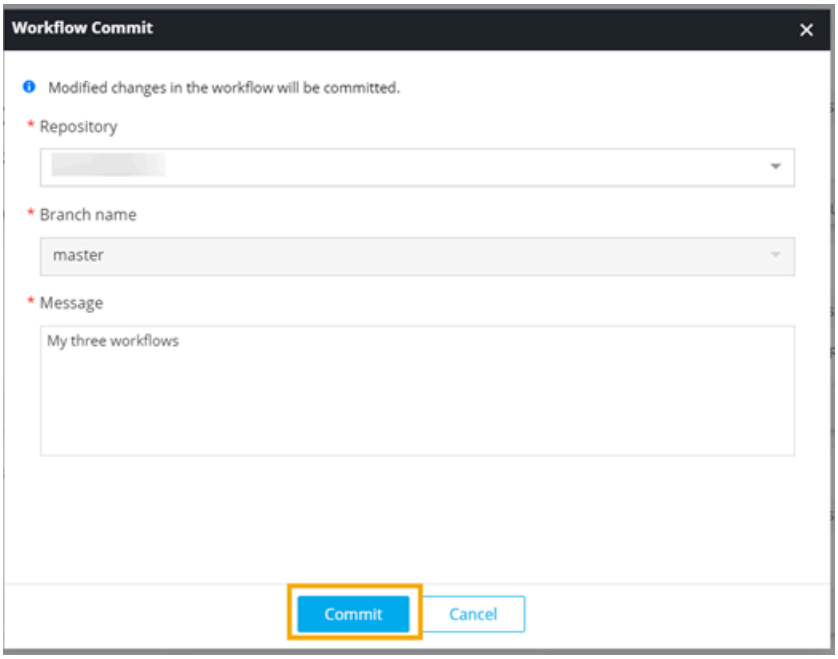
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Committing Workflows from the Workflow Page

1. Go to the Workflow page.
2. On the **Workflow** page, select the workflows to be committed.
3. From the **Source Control** dropdown menu, select **Commit changes**.



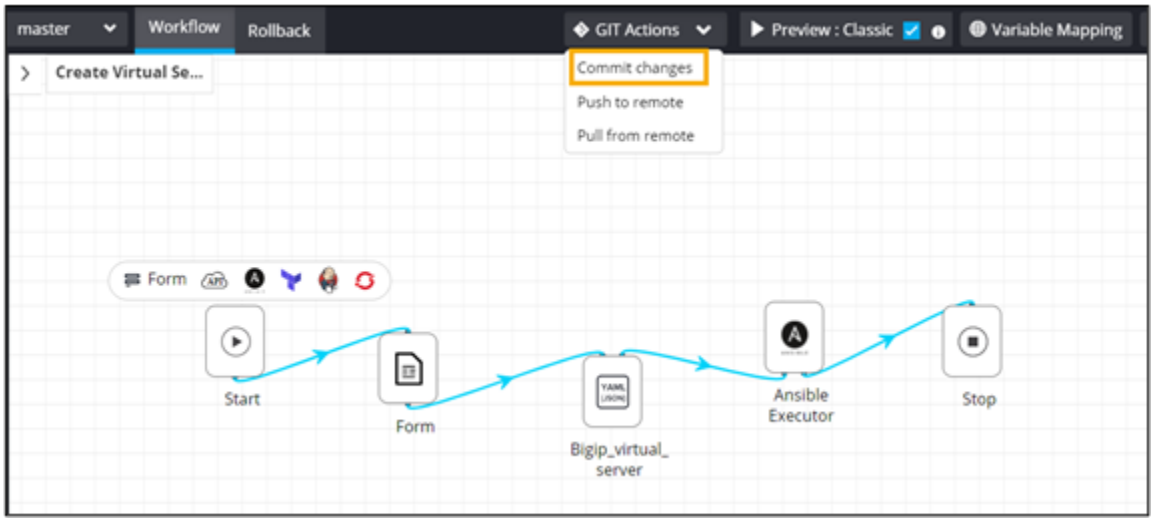
4. In the **Workflow Commit** window, enter a commit **Message** and click **Commit**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Committing Workflows from the Design Canvas

1. Go to the Workflow page.
2. Design a workflow or open an existing workflow that you want to commit to Git.
3. Form the GIT Actions dropdown, select **Commit changes**.

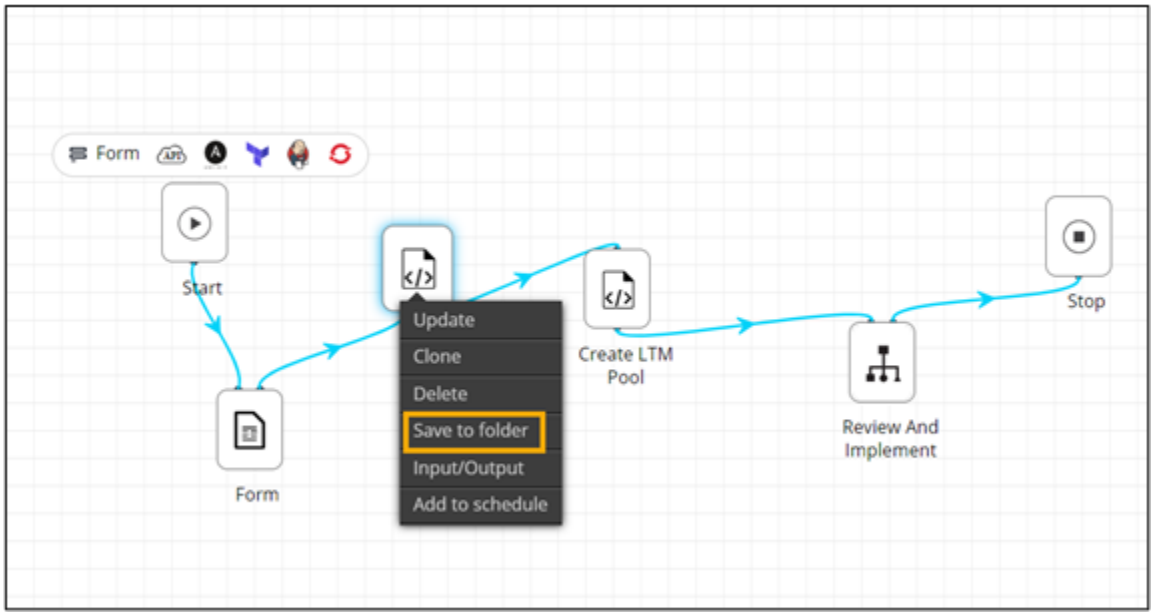


4. In the **Workflow Commit** window, enter a commit **Message** and click **Commit**.

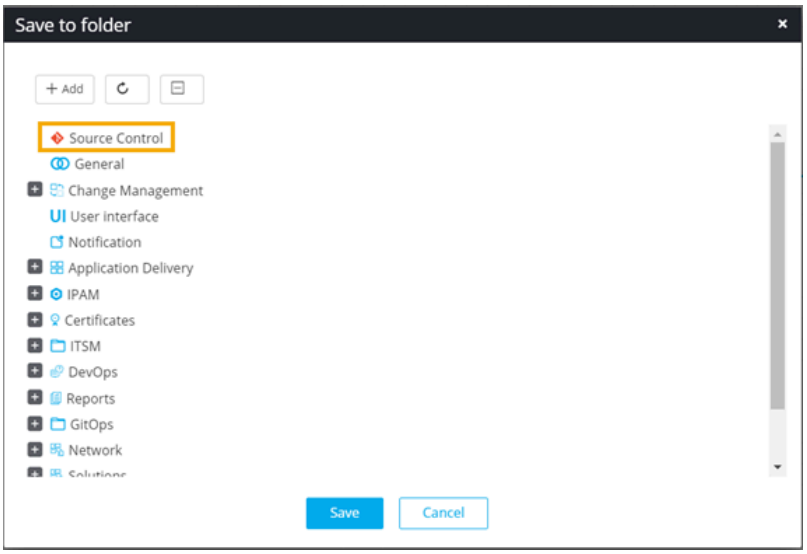
Committing Tasks and Subflows

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

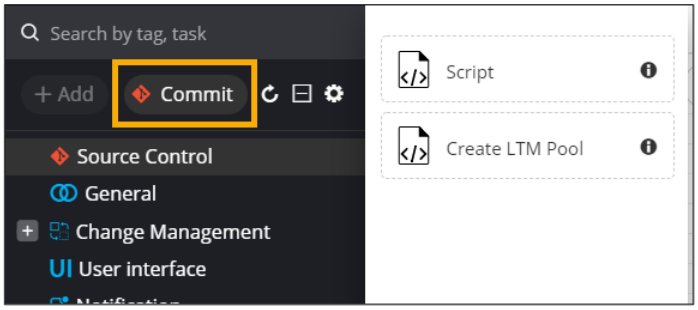


5. In the **Save to folder** window, save the task to the **Source Control** directory.



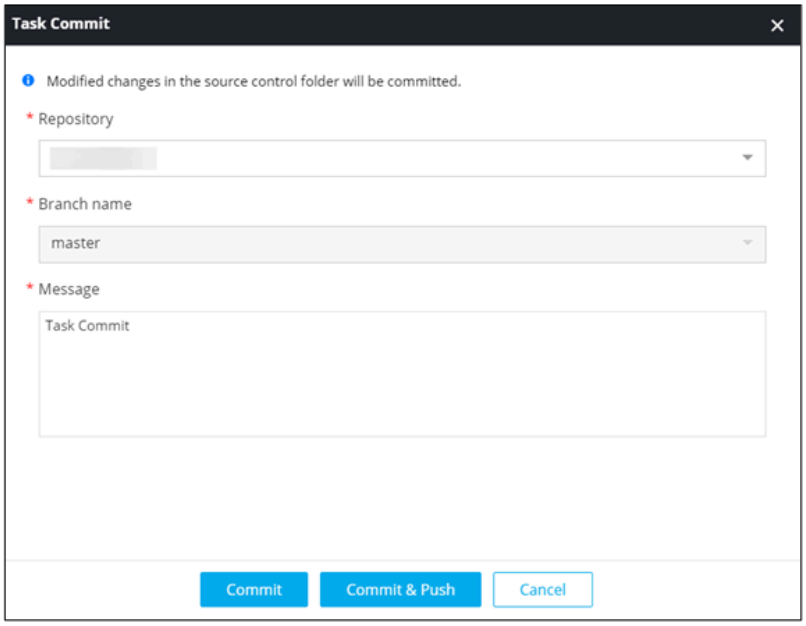
6. Click **Save**.
7. Add the tasks and subflows to be committed to the Source Control Directory.
8. Click **Commit**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

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

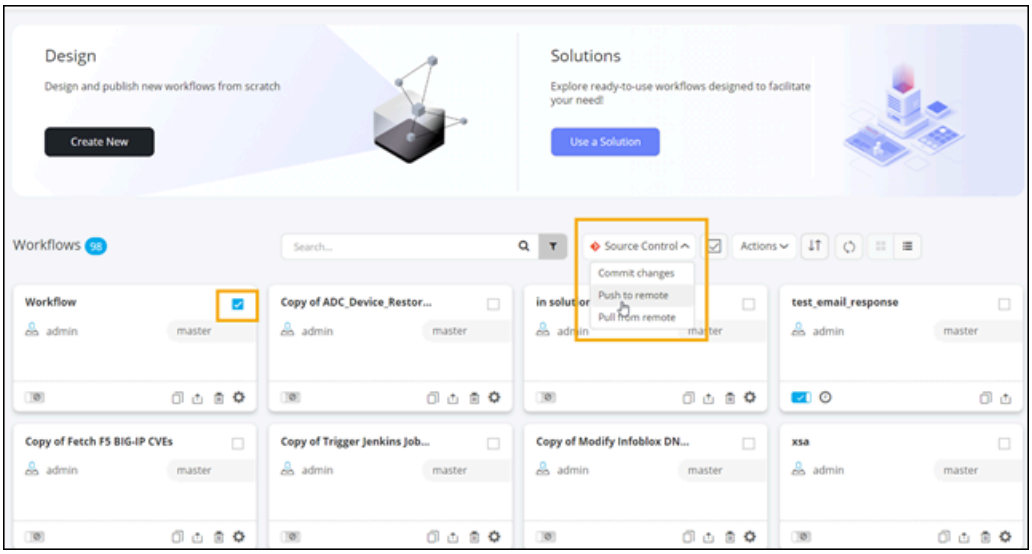


Note: Only the recently added tasks/subflows will be committed.

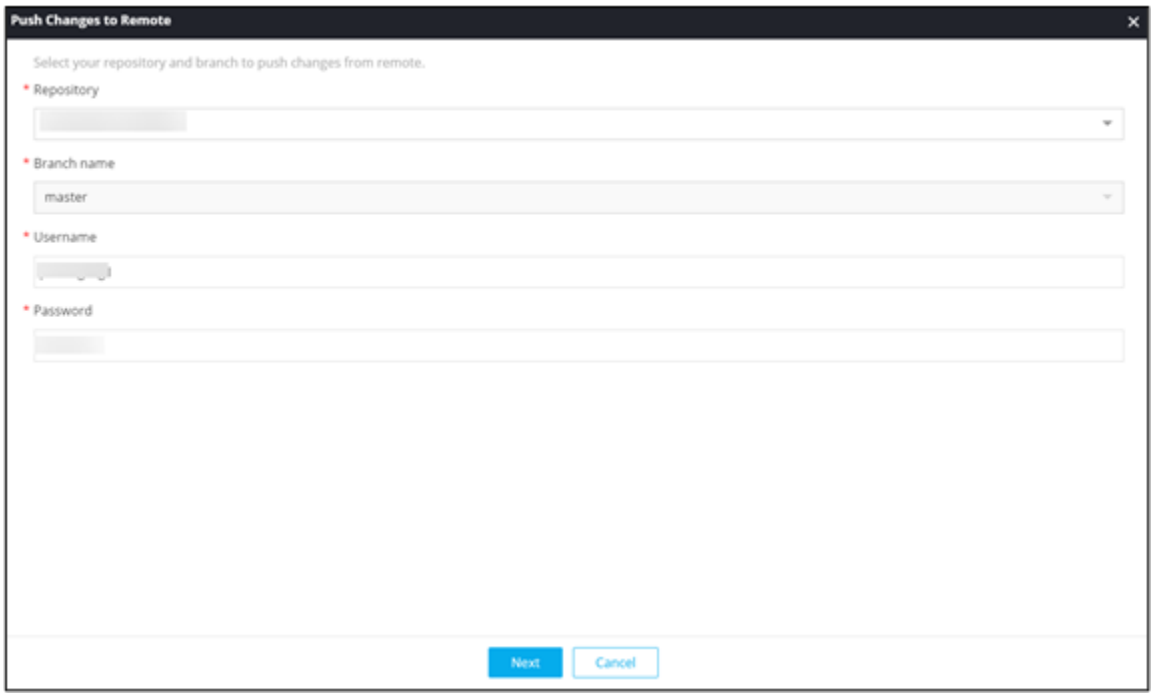
Pushing to a Remote Repository from Workflow Page

1. Go to the Workflow page.
2. On the **Workflow** page, from the **Source Control** dropdown, select **Push to remote**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

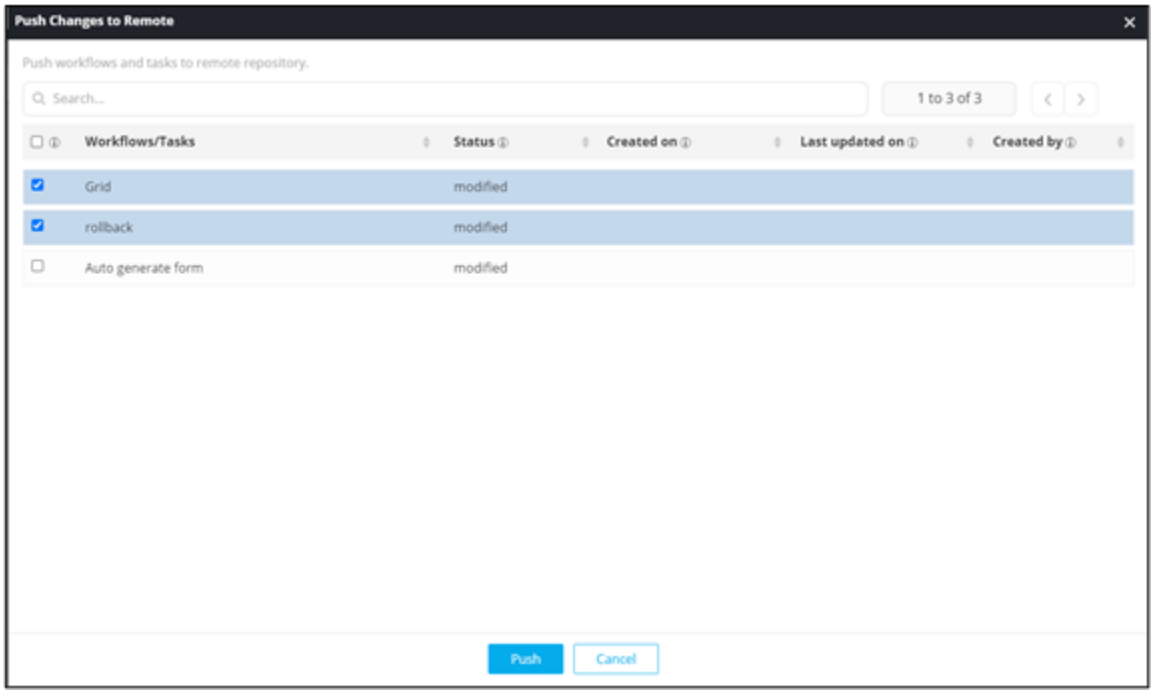


3. In the **Push Changes to Remote** window, enter the **Password**.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

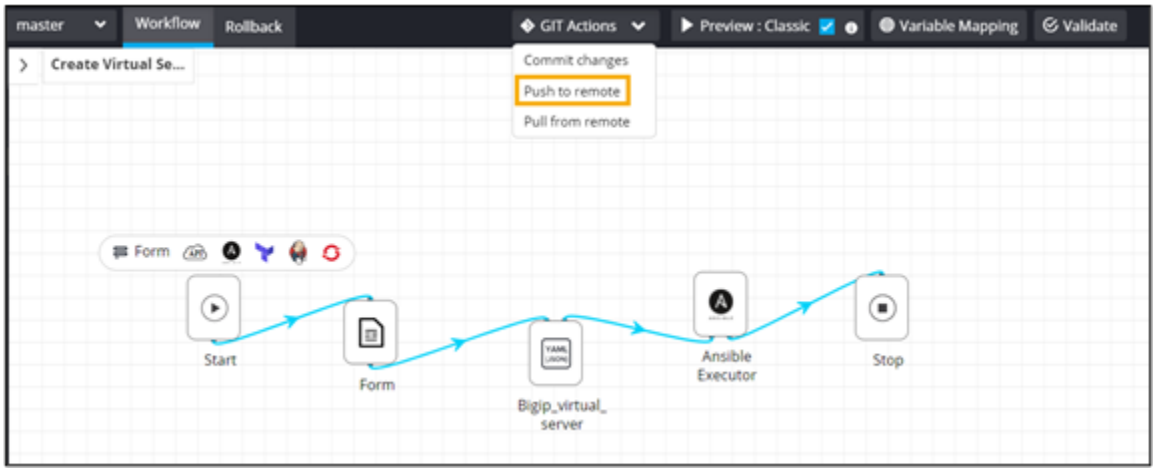


6. Click **Push**.

   The selected workflows/tasks are pushed successfully.

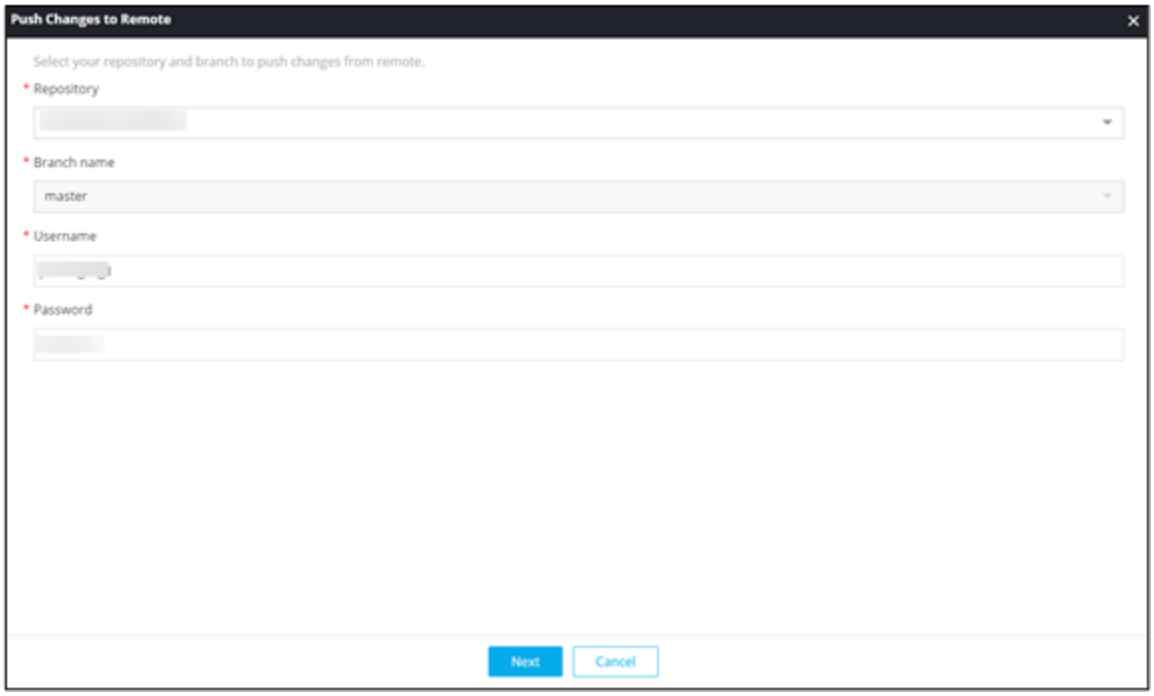
Pushing to a Remote Repository from the Design Canvas

1. Go to the Workflow page.
2. Design a workflow or open an existing workflow that you want to push to Git.
3. From the **GIT Actions** dropdown, select **Push to remote**.

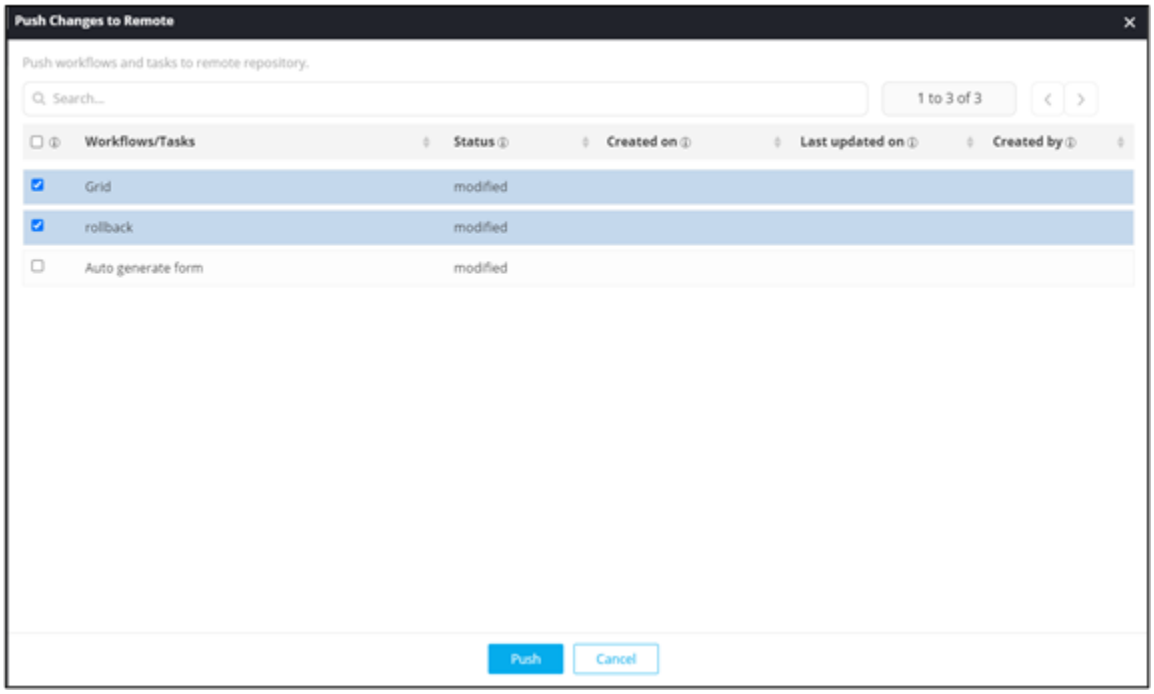


4. In the **Push Changes to Remote** window, enter the **Password**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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



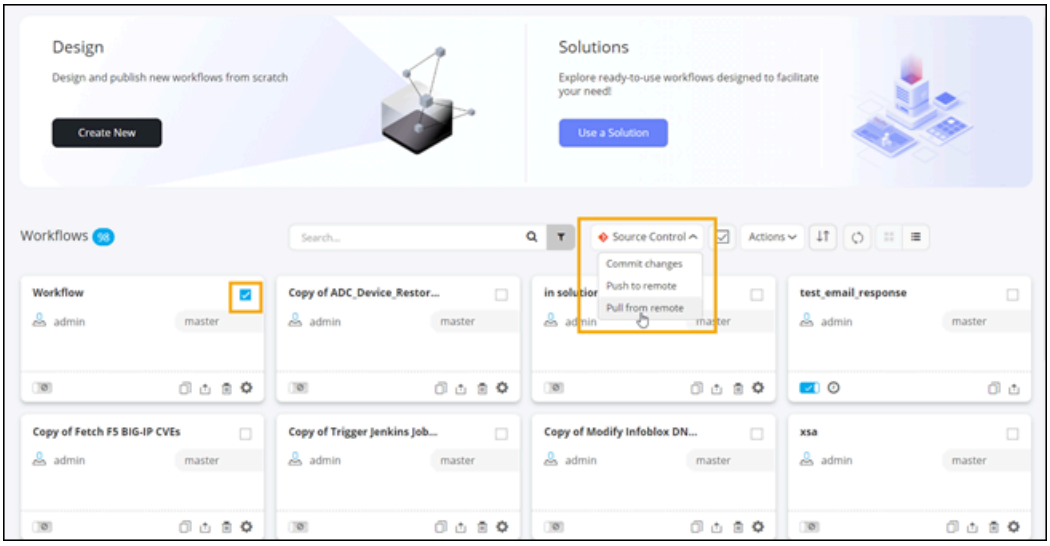
6. Click **Push**.

   The selected workflows/tasks are pushed successfully.

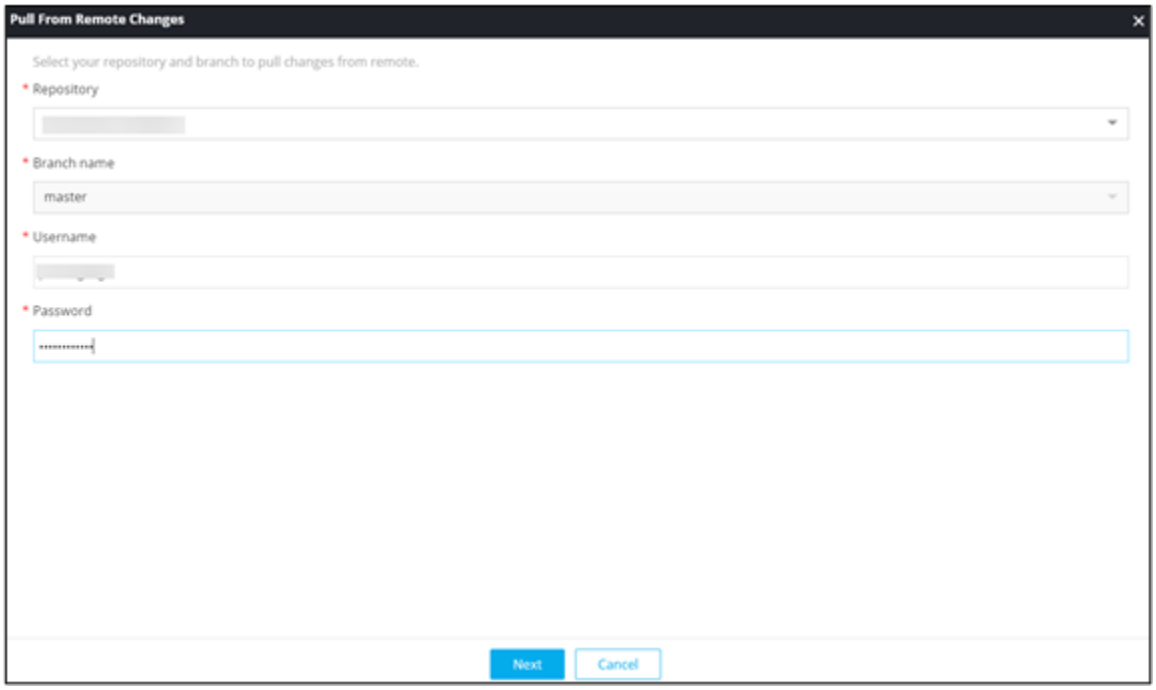
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Pulling from the Repository from Workflow Page

1. Go to the Workflow page.
2. On the **Workflow** page, from the **Source Control** dropdown, select **Pull from remote**.

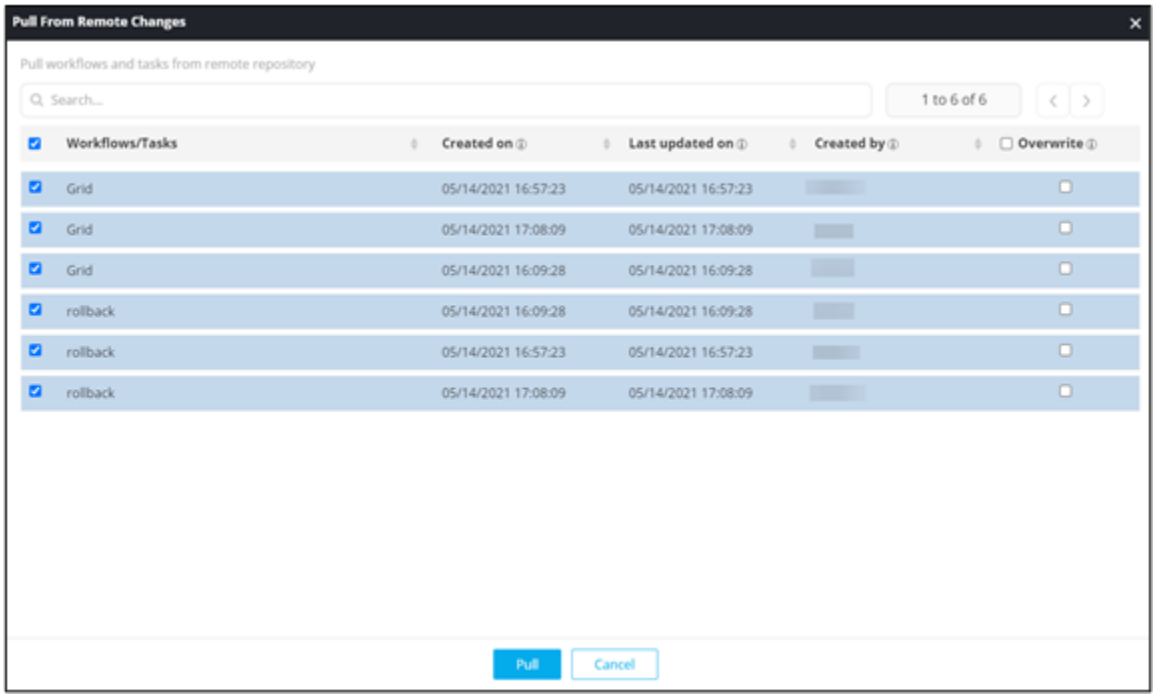


3. In the **Pull From Remote Changes** window, enter the field information.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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

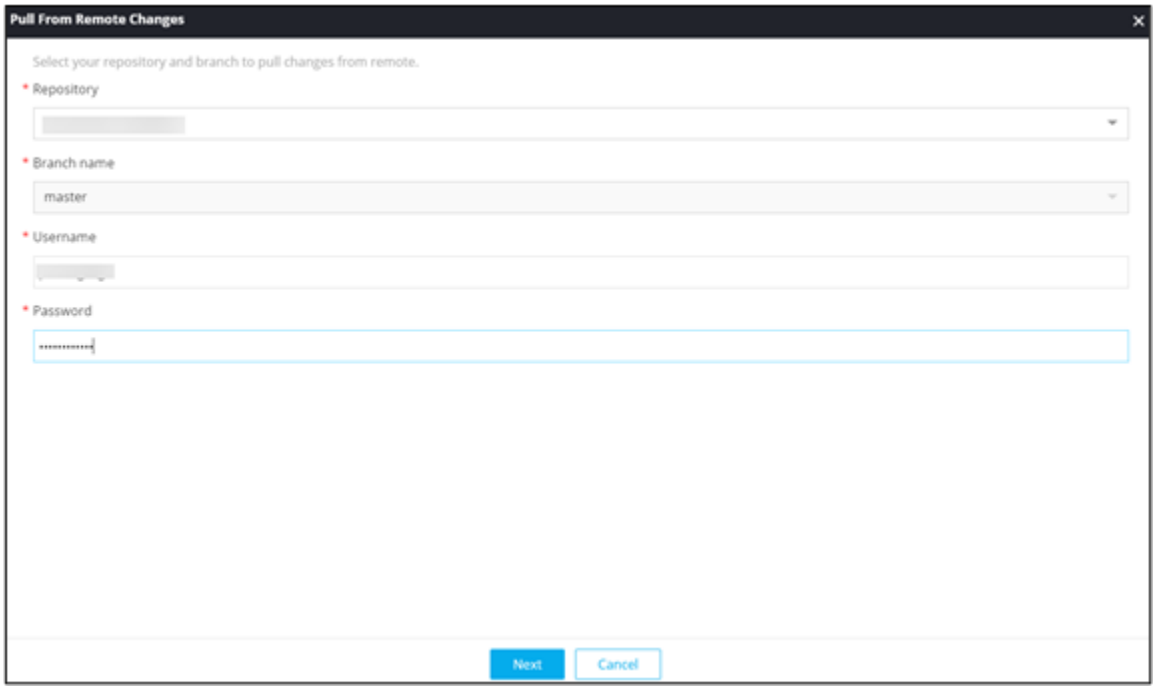


Pulling from the repository from the Design Canvas

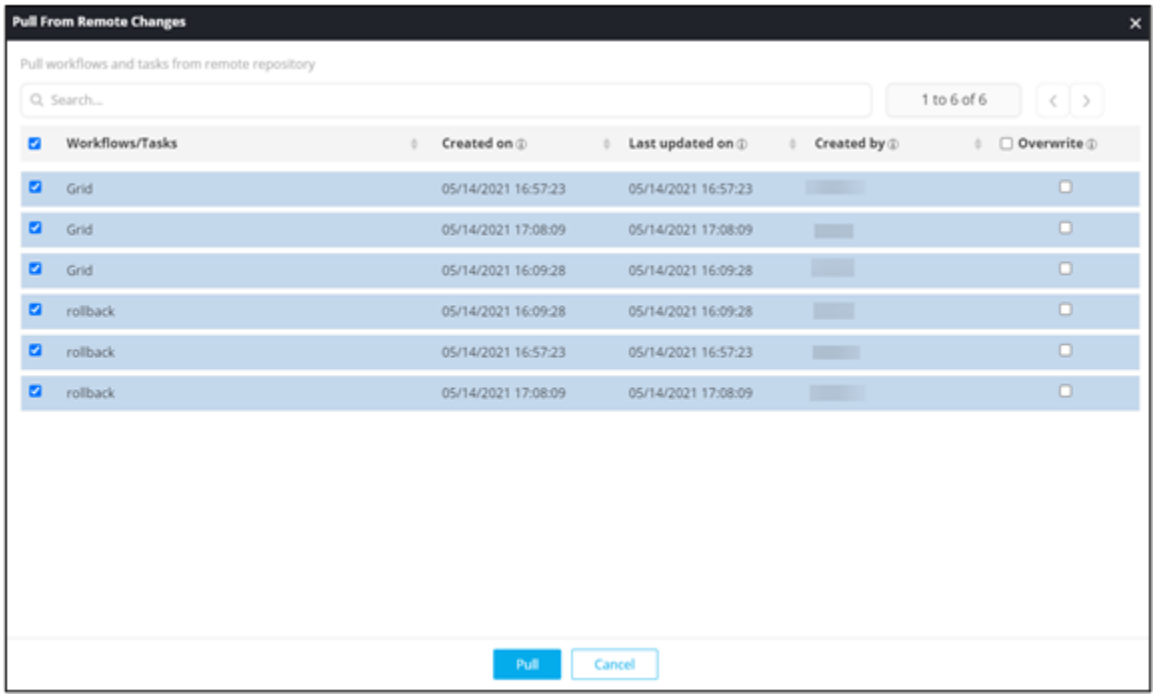
1. Go to the Workflow page.
2. Design a workflow or open an existing workflow into which you want to pull tasks/workflows.
3. From the **GIT Actions** dropdown, select **Pull from remote**.

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Unlinking a Repository

1. Go to the Source Control page.
2. On the **Source Control** page, under **Manage Repositories**, select the repository to be unlinked.

3. Deactivate the repository by turning off the toggle.
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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.
- 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, and Pages.
- Commit and Push actions for workflow(s) and task(s) can be performed from the Workflow page as well from the workflow Design studio.

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

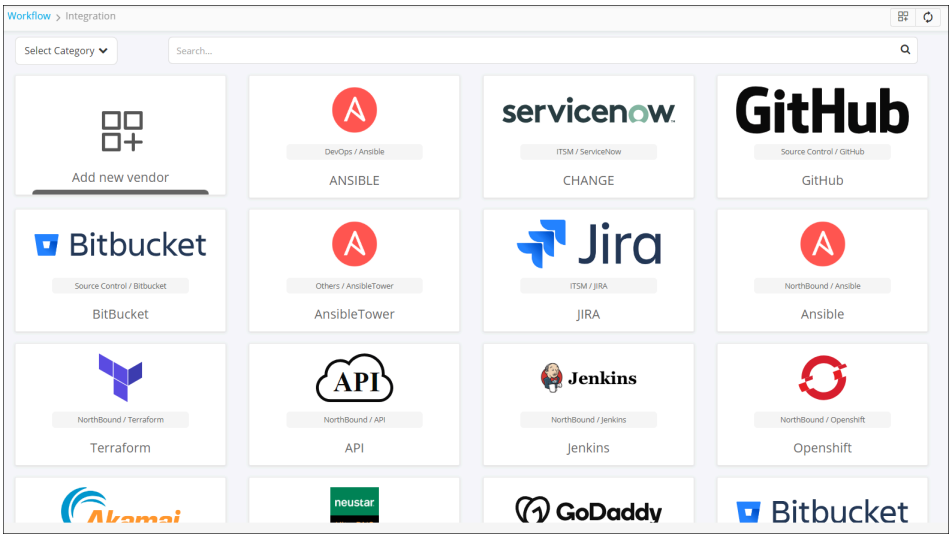
- Option for hiding password in tasks and folder segregation along with filtering field values of exported data in remote is not supported.
- Support is not provided for deleting remote tasks and workflows.

Integration

AppViewX 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 push specific configurations and logs as part of the automation process using variables.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- Accessing the Integration Hub
- ITSM Vendor Configuration
- Jenkins Vendor Configuration
- Ansible Vendor Configuration
- Terraform Vendor Configuration
- Using Variables in Integrations
- Vendor Configurations for PKI+

Accessing the Integration Hub

New Menu	Old Menu
In the Automation module, from the navigation pane on the left, under **WORKFLOW**, select **Integration**. The **Workflow** > **Integration** page is displayed.	1. From the main navigation menu, select **Studio** > **Workflow**. The **Workflow** page is displayed with the **Design** tab open by default. 2. From the navigation pane on the left, select **Integration**. The **Workflow** > **Integration** page is displayed.

Note: For more information on how to switch between menus, click here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

ITSM Vendor Configuration

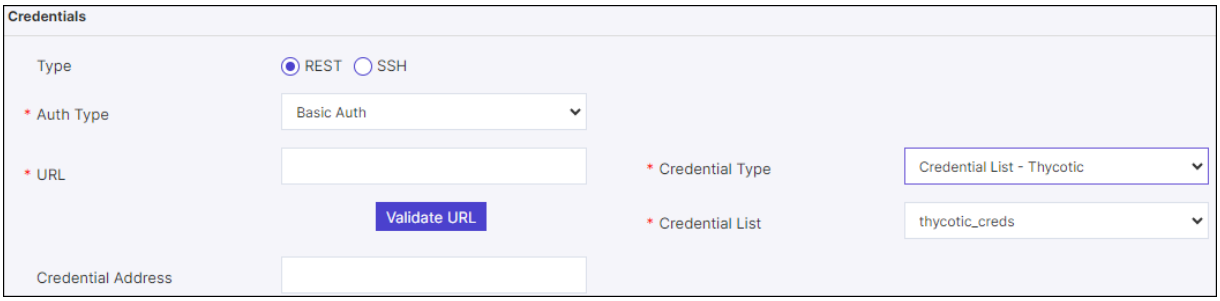
You can configure the relevant ITSM configuration - communication end points, field mappings, closure codes - required to integrate a workflow automation flow with an IT Service Management (ITSM) tool.

- 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
  - 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.

- Configuring a ServiceNow Instance
- Checking ServiceNow REST API details
- ITSM JSON Configuration

Configuring a ServiceNow Instance

1. Go to the Integration Hub.
2. On the **Workflow** > **Integration** page, click **servicenow/change**.
3. On the **Vendor Configuration** page that is displayed, under **Credentials**, enter the required field information.



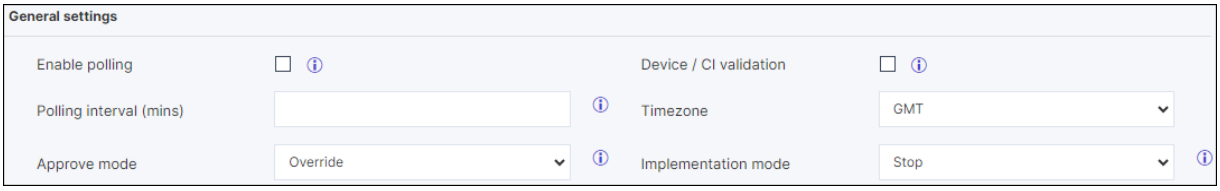
From the **Auth type** field, if you select **Basic Auth** and **Credentials** you can select the **Credential Type** from the dropdown and enter the relevant details to support credential authentication.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- In the **Credential Type** field, if *Manual* is selected, the enter the **Username** and **Password**.
- In the **Credential Type** field, if *Credentail List - XYZ* is selected, then in the **Credential List** field, a list of credentials from credential store will be displayed.
  - If the **Credential Type** = *Credentail List - CyberArk*, *Credentail List - BeyondTrust* an additional field **Credential Address** is displayed. It uniquely identifies the credential list.

> **Note:** The credentials must be added in the respective **Device Inventory**.

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

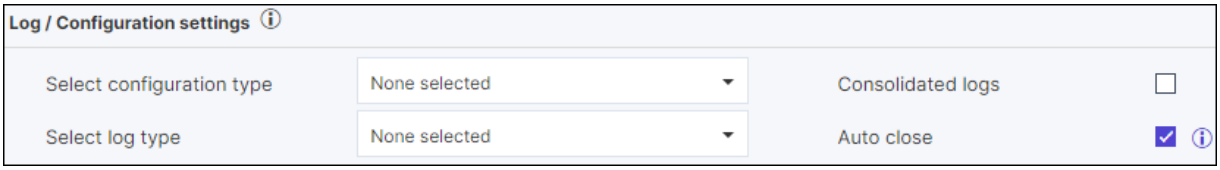


Field descriptions for General Settings

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 © 2025 AppViewX, Inc. All Rights Reserved.

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



Field descriptions for Log/Configuration

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. Under **Configurations**, turn on the **Enable config** toggle and define relevant API parameters in the JSON configurator that is displayed.

   Sample Schema:

```
{
 "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"
    },
    "apiListToCallAfter": [
     "getDeviceList"
    ],
    "name": "getTicket",
    "method": "GET"
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
  },
  "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"
  }
 },
 "validation": {
  "state": "approved"
 },
 "tooltipData": {
  "Affected CI's": "deviceList",
  "Start Date": "startDate",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
  "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"
 }
}
}
```

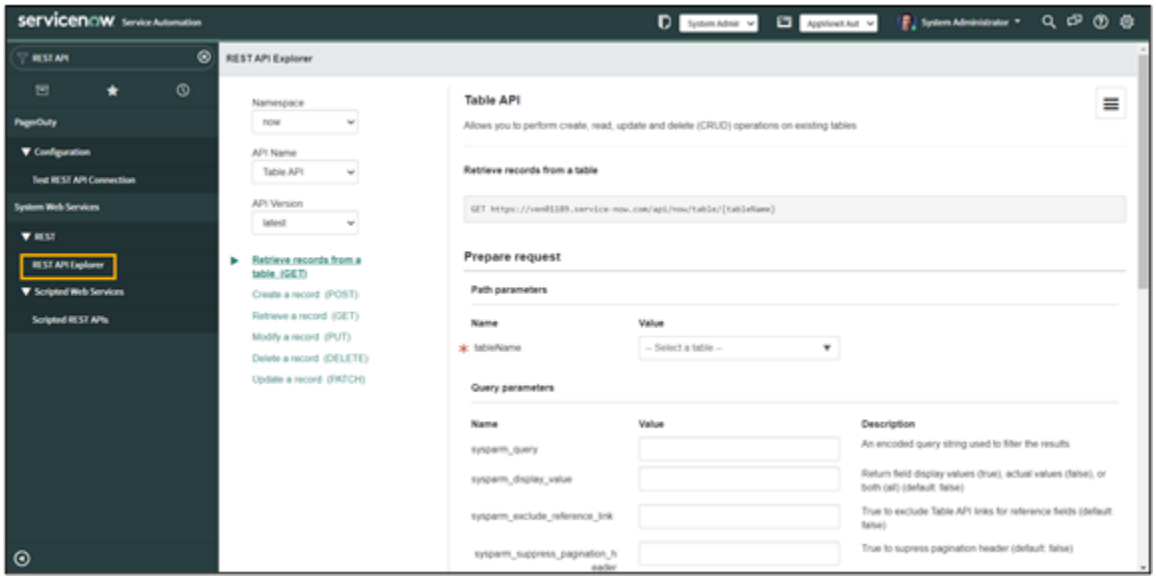

7. Click **Add**.

Checking ServiceNow REST API details

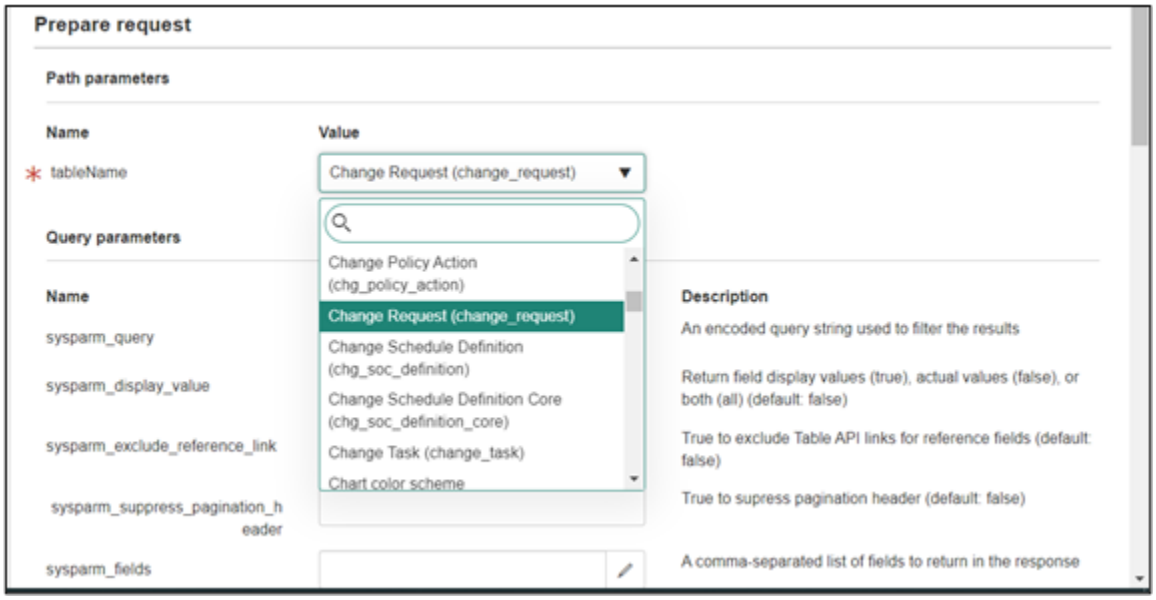
To retrieve details using REST API explorer for a given Change ticket on ServiceNow:

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

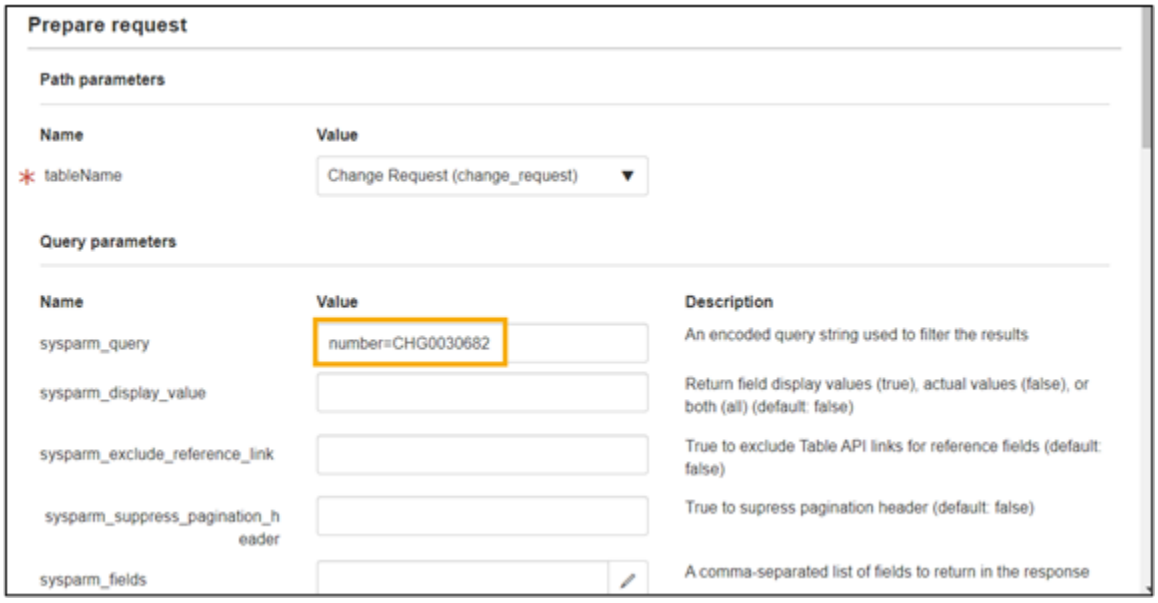


3. Select **tableName** value from the dropdown list.



4. Enter the value for **sysparm_query** (number=<change ticket>).
   For example, number=CHG0030682

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

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


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"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",
"work_notes_list": "",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"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",
"closed_by": {
```

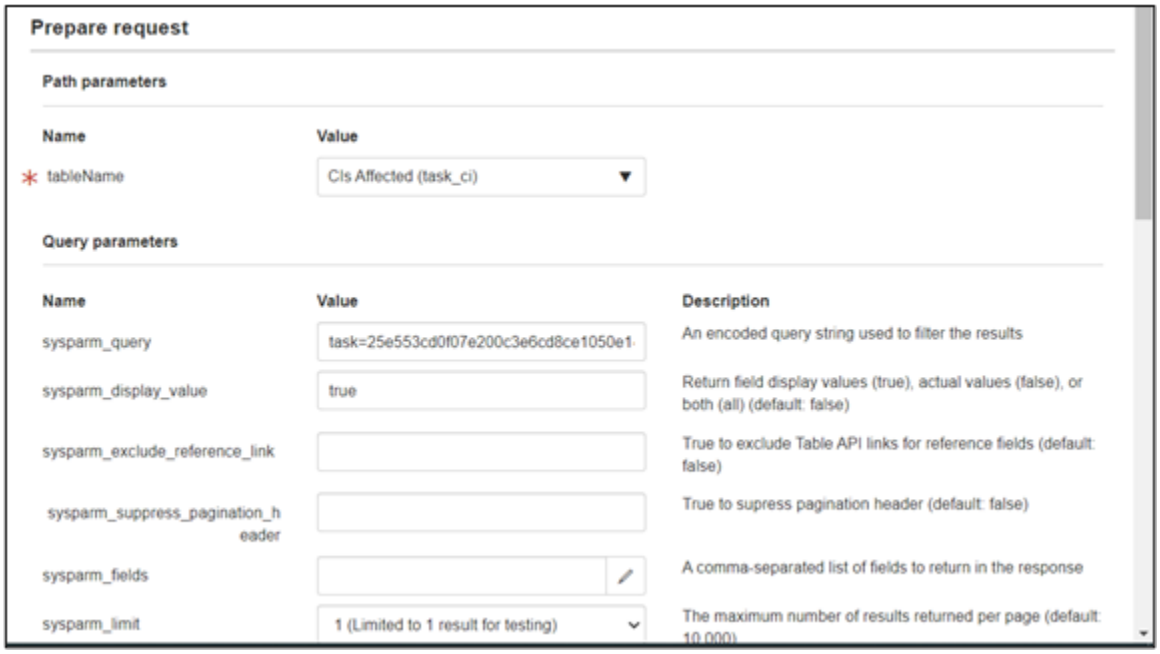

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
      "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"
  }
 ]
}
```


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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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",
   "sys_updated_on": "2016-11-11 07:08:23",
   "sys_tags": "",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "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
- apiListToCallAfter
- apiListToCallBefore.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• ITSM APIs
• URL and Response Data Mapping
• Response Data Mapping
• Payload Data Mapping
• Validation Parameters
• Tooltip Data Configuration
• Mandatory Headers
• Authorization Configuration

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 © 2025 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 © 2025 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"
 },
```

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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

```
{
 "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 © 2025 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.

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": {
   "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.

Validation Parameters

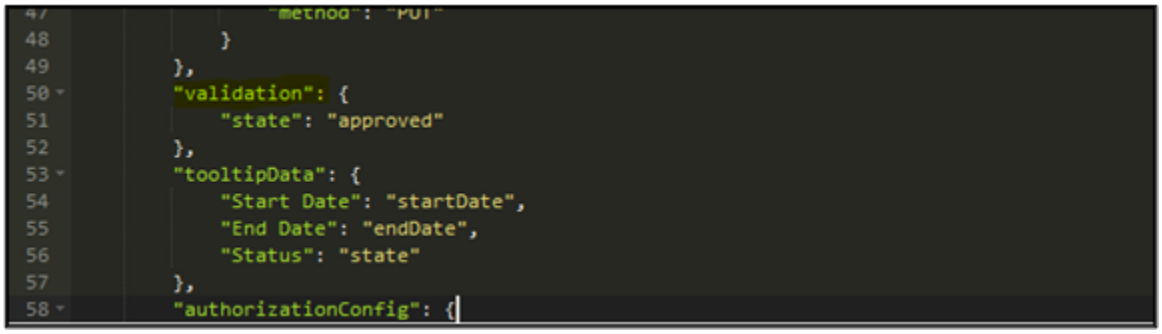
This is a set of JSON key value pairs that define what attributes of a change ticket needs to be validated within AppViewX. Any custom attributes can be defined as part of the validation. The values on the left

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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 configuration:

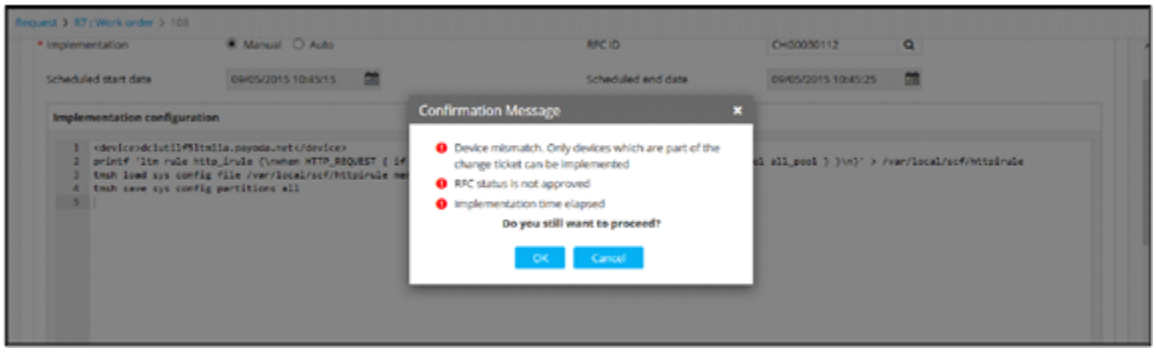



```
"name": "updateTicket",
  "method": "PUT"
 }
 },
 "validation": {
 "state": "approved"
 },
 "tooltipData": {
 "Start Date": "startDate",
 "End Date": "endDate",
```

• 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



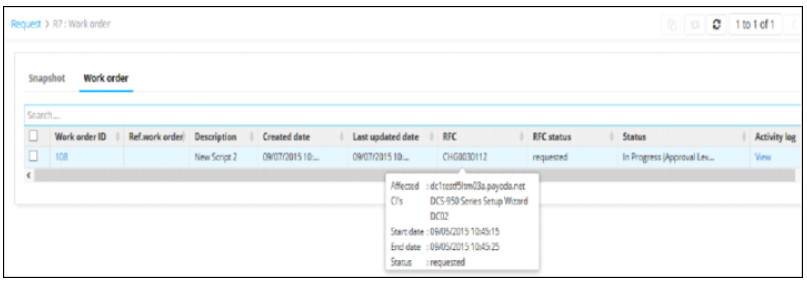
Tooltip Data Configuration

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",
        "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:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

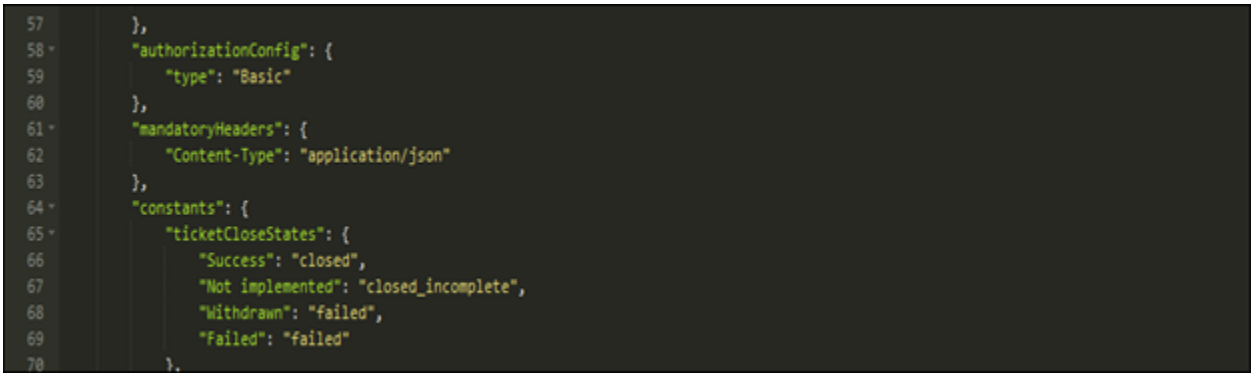
```
"mandatoryHeaders":{
      "Content-Type":"application/json"
  }
```

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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"
  }
```

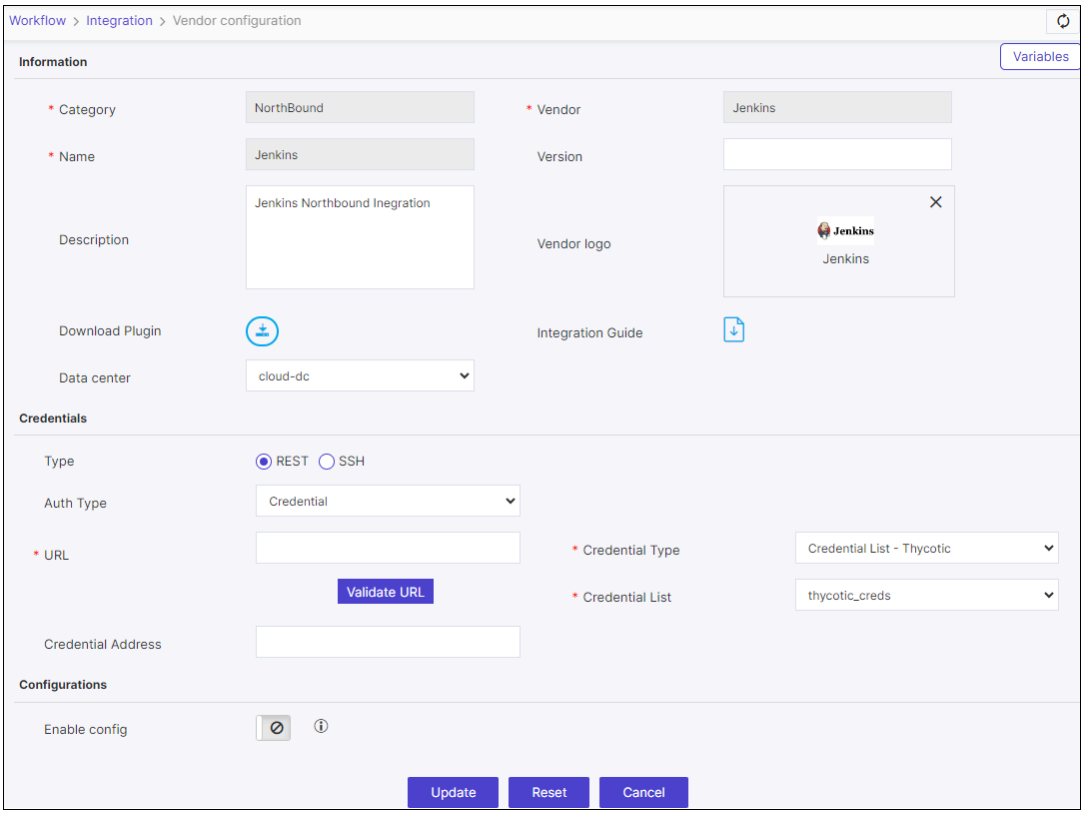
Example: Ticket closure states in string format

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

Jenkins Vendor Configuration

1. Go to the Integration Hub.
2. On the **Workflow** > **Integration** page, click **Jenkins**.
3. On the **Vendor configuration** page, enter the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: Under Authentication type **Basic Auth** and **Credentials** you can select the credential type from the drodown and enter the relevent details to support credential authentication.

From the **Auth type** field, if you select **Basic Auth** and **Credentials** you can select the **Credential Type** from the dropdown and enter the relevant details to support credential authentication.

- In the **Credential Type** field, if *Manual* is selected, the enter the **Username** and **Password**.
- In the **Credential Type** field, if *Credentail List - XYZ* is selected, then in the **Credential List** field, a list of credentials from credential store will be displayed.
  - If the **Credential Type** = *Credentail List - CyberArk*, *Credentail List - BeyondTrust* an additional field **Credential Address** is displayed. It uniquely identifies the credential list.

Note: The credentials must be added in the respective **Device Inventory**.

4. Click **Save**.

Note: For more information on Jenkins - Visual Workflow Northbound integration, click here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Ansible Vendor Configuration

AppViewX allows you to integrate with Ansible in order to automate and orchestrate the network.

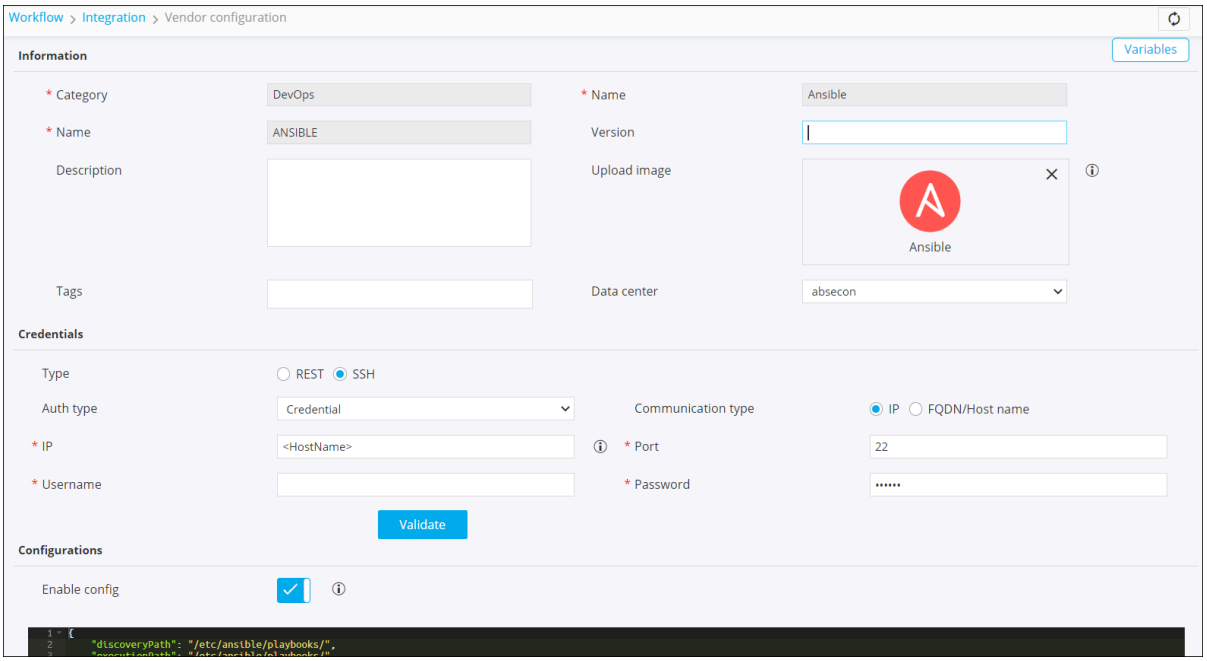
- Provision to integrate with an existing Ansible instance.
- Provision to use the Ansible executor task when there is a need for automating network configurations through Ansible as the southbound.
- Provision to Discover and Import playbooks from a specific path within the visual workflow studio.
- Drag & drop playbooks and automate using Ansible modules via visual workflow.

- Configuring an Ansible Instance
- Discovering Playbooks
- Getting Device Credentials Dynamically
- Ansible Executor

Configuring an 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.

1. Go to the Integration Hub.
2. On the **Workflow** > **Integration** page, click **Ansible**.
3. On the **Vendor configuration** page, enter the required field information.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

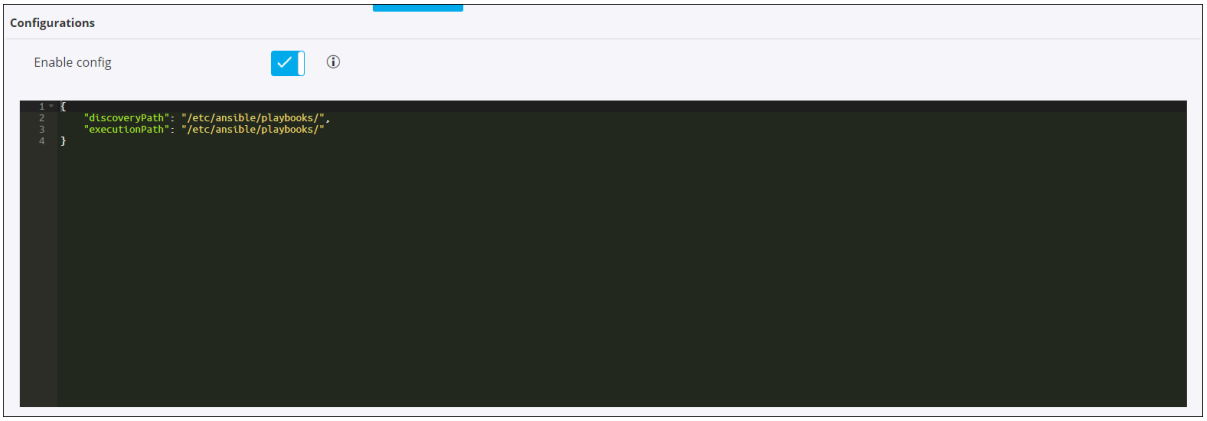
> **Note:** Under Authentication type **Basic Auth** and **Credentials** you can select the credential type from the drodown and enter the relevent details to support credential authentication.

From the **Auth type** field, if you select **Basic Auth** and **Credentials** you can select the **Credential Type** from the dropdown and enter the relevant details to support credential authentication.

- In the **Credential Type** field, if *Manual* is selected, the enter the **Username** and **Password**.
- In the **Credential Type** field, if *Credentail List - XYZ* is selected, then in the **Credential List** field, a list of credentials from credential store will be displayed.
  - If the **Credential Type** = *Credentail List - CyberArk*, *Credentail List - BeyondTrust* an additional field **Credential Address** is displayed. It uniquely identifies the credential list.

> **Note:** The credentials must be added in the respective **Device Inventory**.

4. Under **Configurations**, turn on the **Enable config** toggle and define the folder/path from where existing playbooks can be discovered within the workflow studio.



5. Click **Add**.

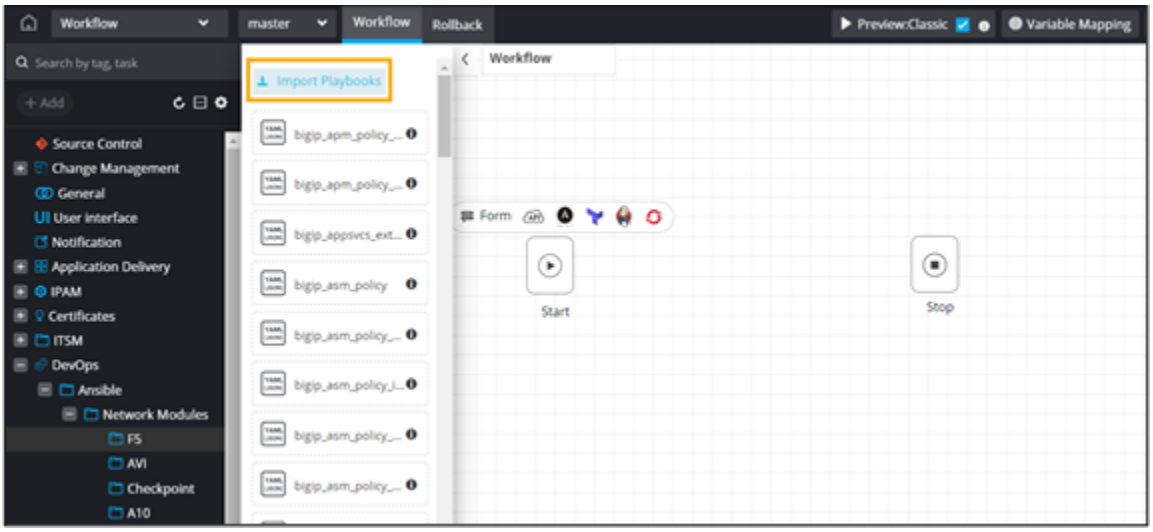
> **Note:** For more information on Ansible - Visual Workflow Northbound integration, click here.

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.

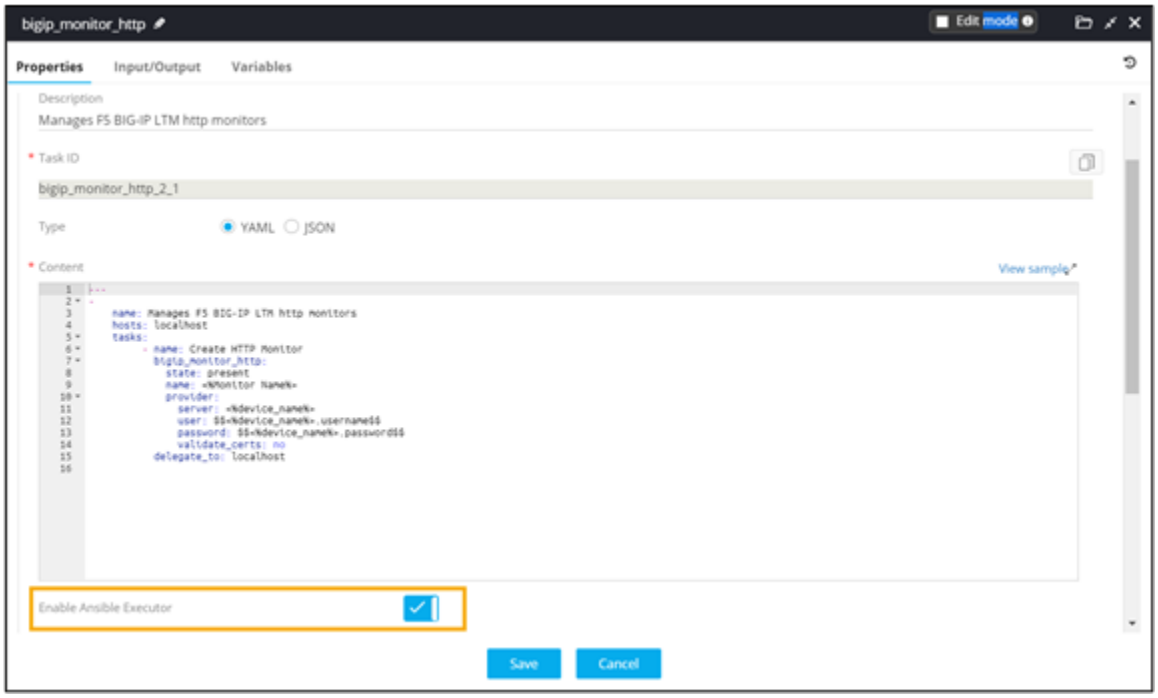
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Go to the Workflow page.
2. Design a workflow.
3. From the menu on the left, under **DevOps**, select **Ansible** > **Network Modules** > **F5**.
4. From the **F5** folder, click **Import Playbooks**.

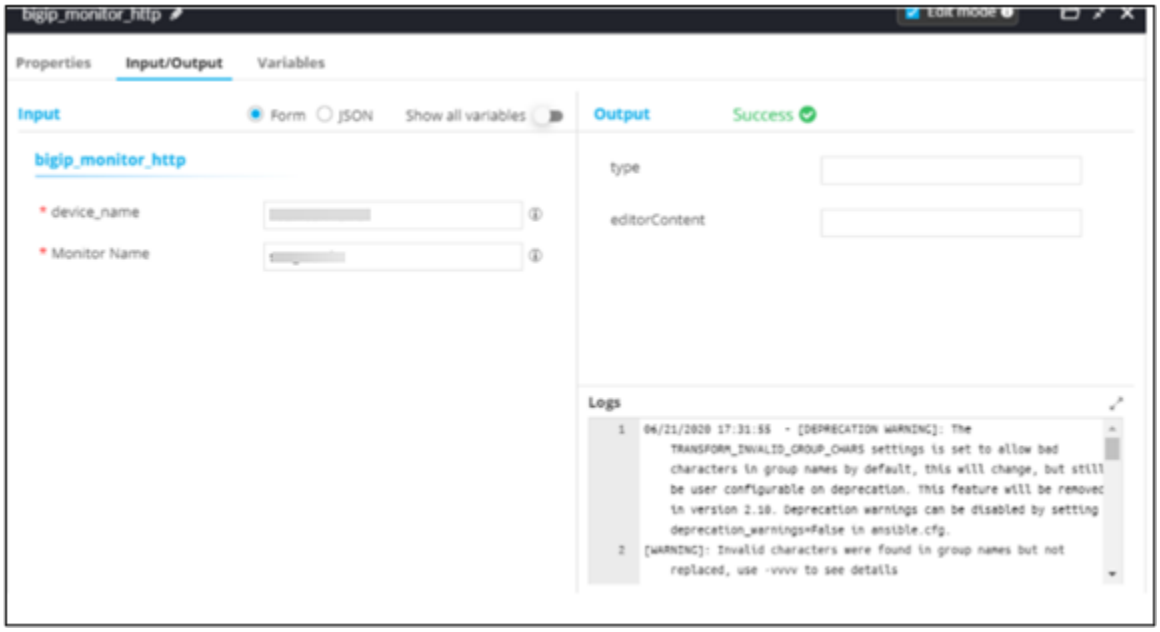


5. In the **Import playbooks** window, select import **Type**.
   - **Upload file**: Upload playbooks from a local folder.
   - **Import playbooks**: Discover playbooks from Ansible folder.
6. Save playbooks to a custom folder.
7. Drag and drop the playbooks into the workspace.
8. In the task window, under **Properties**, to enable the Ansible instance, turn on the toggle.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

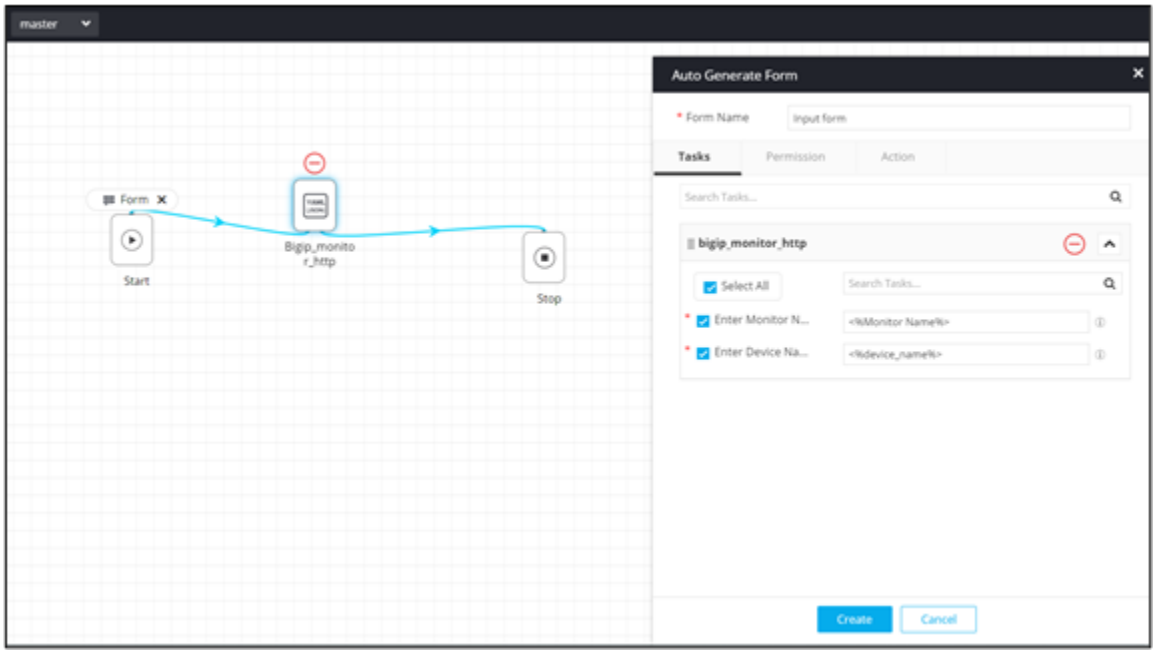


9. Under **Input/Output**, enter the **Input/Output** and **Variables** details.
10. Execute the task to view the execution log.



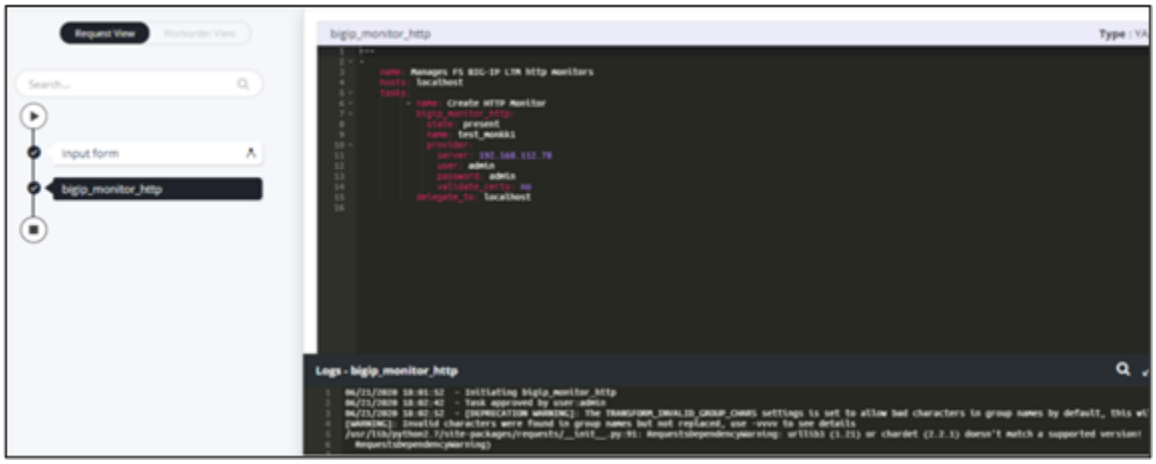
11. Auto generate a self-service form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



12. Connect the workflow tasks and enable the workflow.
13. Trigger the workflow.

    Workflow is executed.



14. To view the Request activity log, on the Request :: View/Run page, under the **Activity** column, click **View**.

    A **Summary** of the workflow request execution is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.

To get the device credentials dynamically during the workflow execution process:

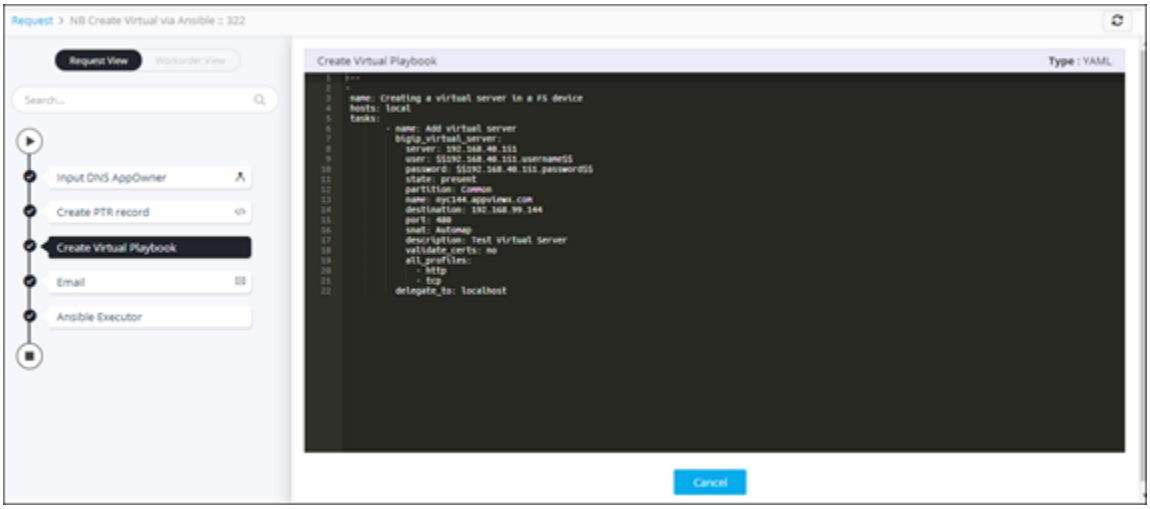
1. Go to the Integration Hub.
2. Design a new workflow.
3. Drag and drop relevant tasks.
4. From the **User Interface** section in the left menu, drag and drop the **YAML** task.
5. Add input data in the YAML format.
6. Enter the following syntax to get the device credentials dynamically.

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

7. Connect the workflow tasks and enable the workflow.
8. Trigger the workflow.

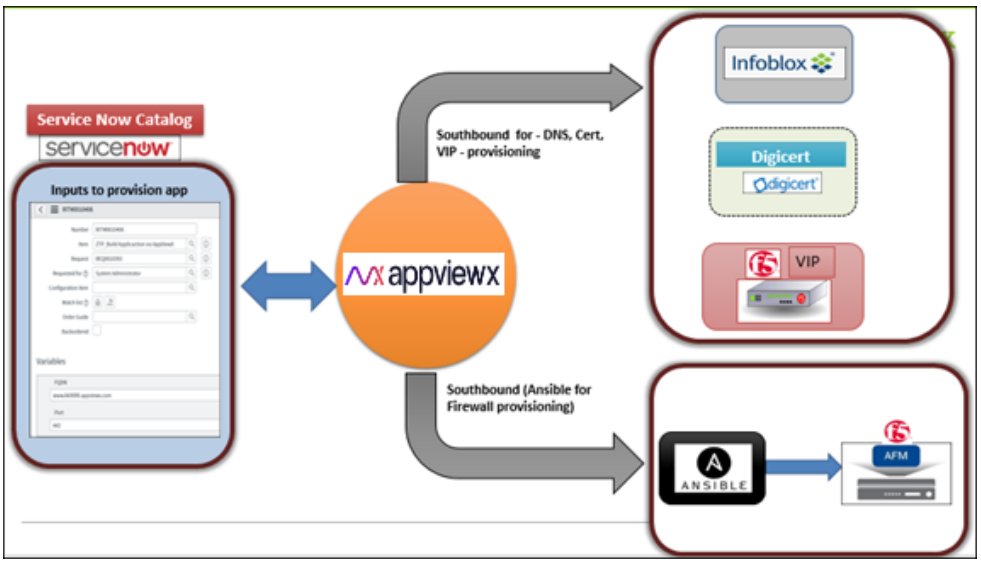
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Device details are fetched.



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.

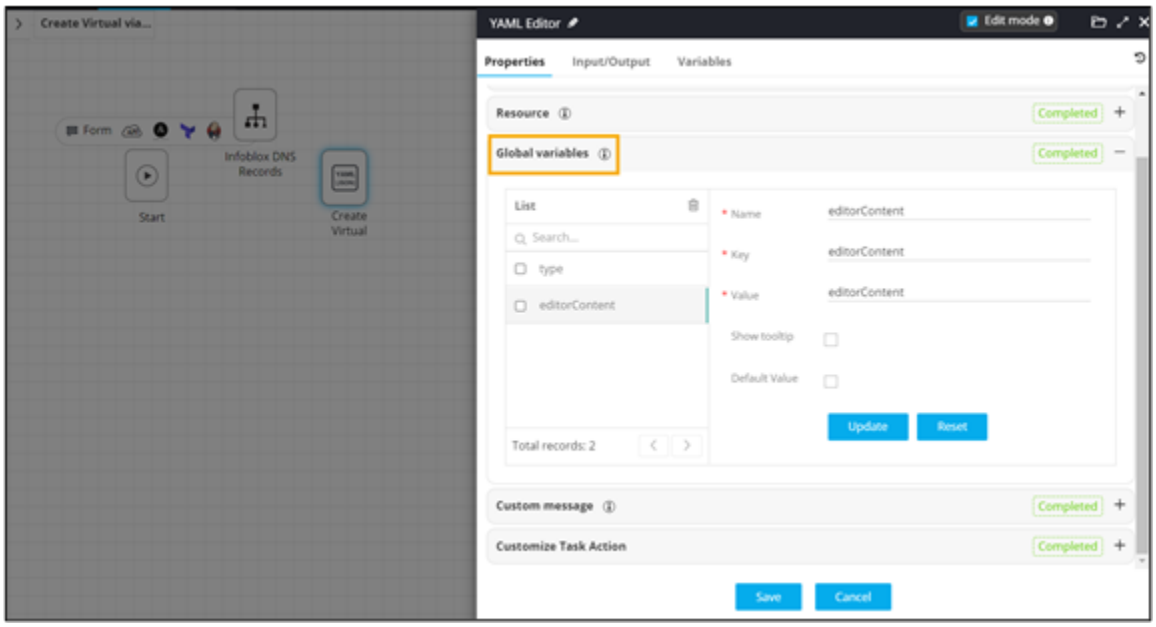


To connect to an Ansible instance, discover and reuse a BIG-IP playbook and integrate with Infoblox IPAM via Visual Workflow:

1. Configure an Ansible instance.
2. Go to the Workflow page.

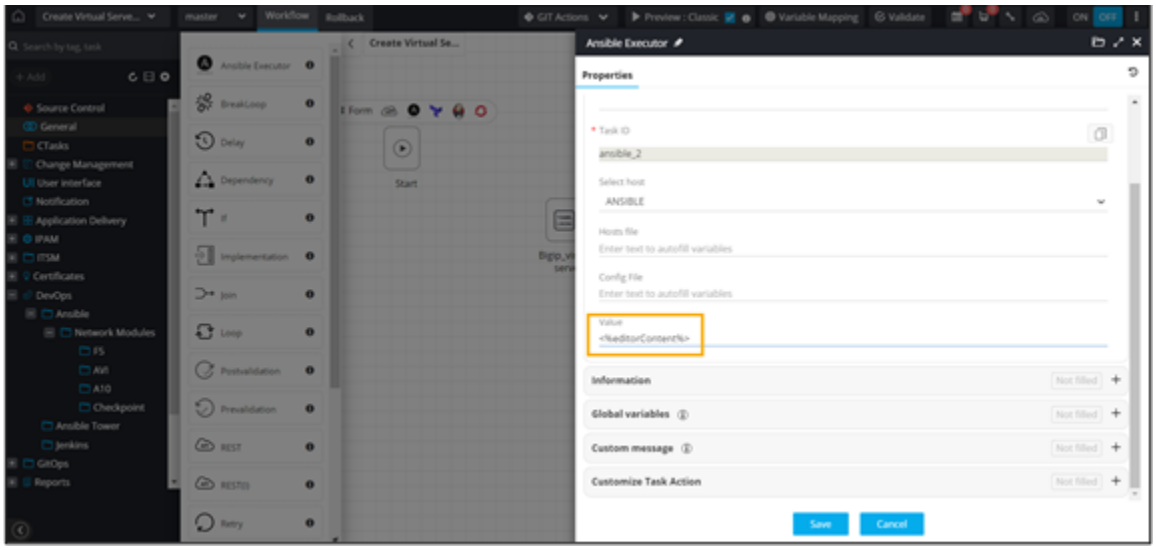
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Design a new workflow.
4. In the design workspace, from the **DevOps** folder in the left menu, select **Ansible** > **Network Modules** > **F5**.
5. From the **F5** folder in the left menu, drag and drop the **BigIP_Virtual_Server** task into the workspace.
6. 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.



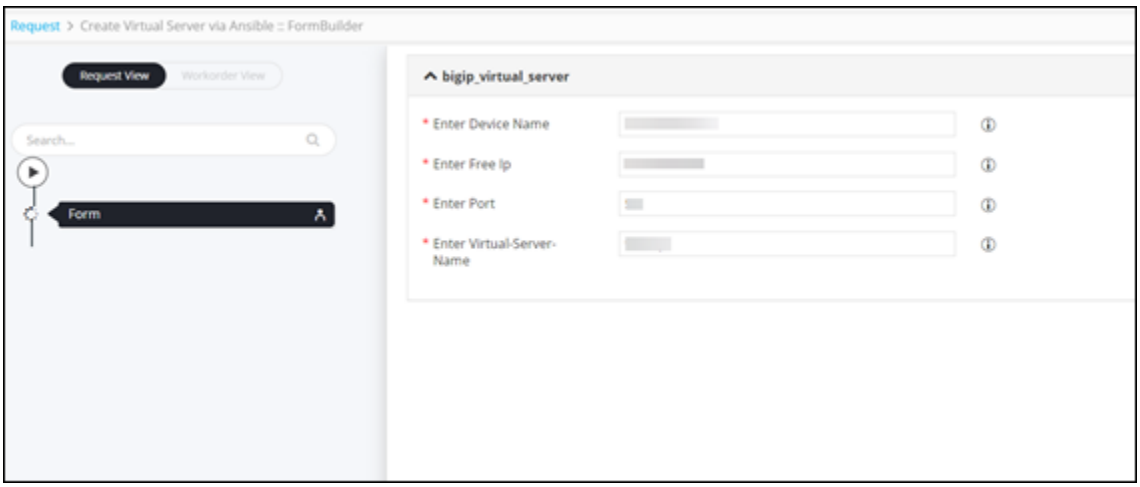
7. From the **General** section, drag and drop the **Ansible Executor** task.
8. In the **Ansible Executor** task window, under **Properties**, select the Ansible host on which implementation is to be done.
9. Map the global variable from the YAML task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



10. Auto-generate a self-service form.
11. Connect the workflow tasks and enable the workflow.
12. Trigger the workflow.

    The workflow execution page is displayed with the workflow inputs requested at the first stage.
13. Enter the input details in the Form.



14. Click **Next**.
15. Submit the YAML task.

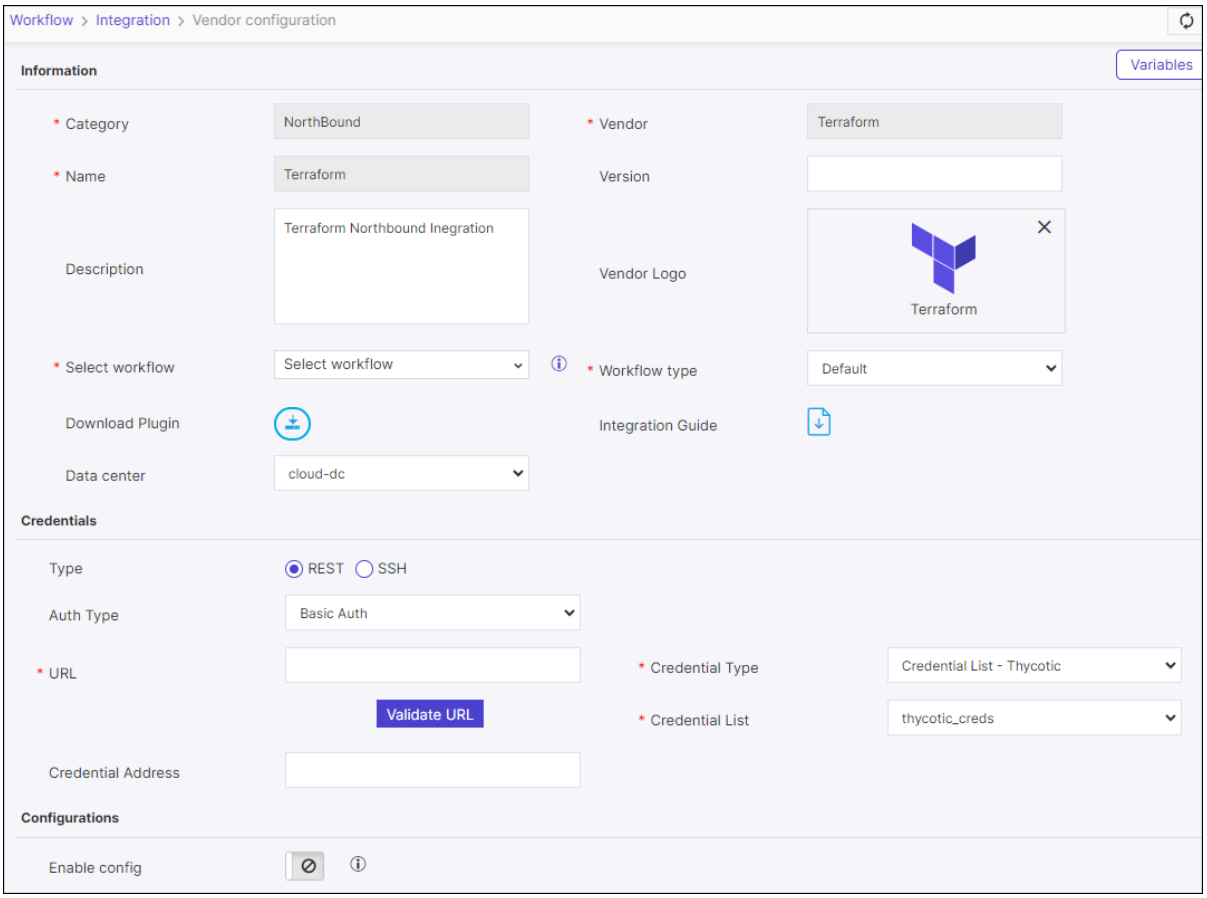
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Ansible Executor task completed.

Terraform Vendor Configuration

1. Go to the Integration Hub.
2. On the **Workflow** > **Integration** page, click **Terraform**.
3. On the **Vendor configuration** page, enter the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: Under Authentication type **Basic Auth** and **Credentials** you can select the credential type from the drodown and enter the relevent details to support credential authentication.

From the **Auth type** field, if you select **Basic Auth** and **Credentials** you can select the **Credential Type** from the dropdown and enter the relevant details to support credential authentication.

- In the **Credential Type** field, if *Manual* is selected, the enter the **Username** and **Password**.
- In the **Credential Type** field, if *Credentail List - XYZ* is selected, then in the **Credential List** field, a list of credentials from credential store will be displayed.
  - If the **Credential Type** = *Credentail List - CyberArk*, *Credentail List - BeyondTrust* an additional field **Credential Address** is displayed. It uniquely identifies the credential list.

Note: The credentials must be added in the respective **Device Inventory**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. Click **Save**.

> **Note:** For more information on Terraform - Visual Workflow Northbound integration, click here.

Using Variables in Integrations

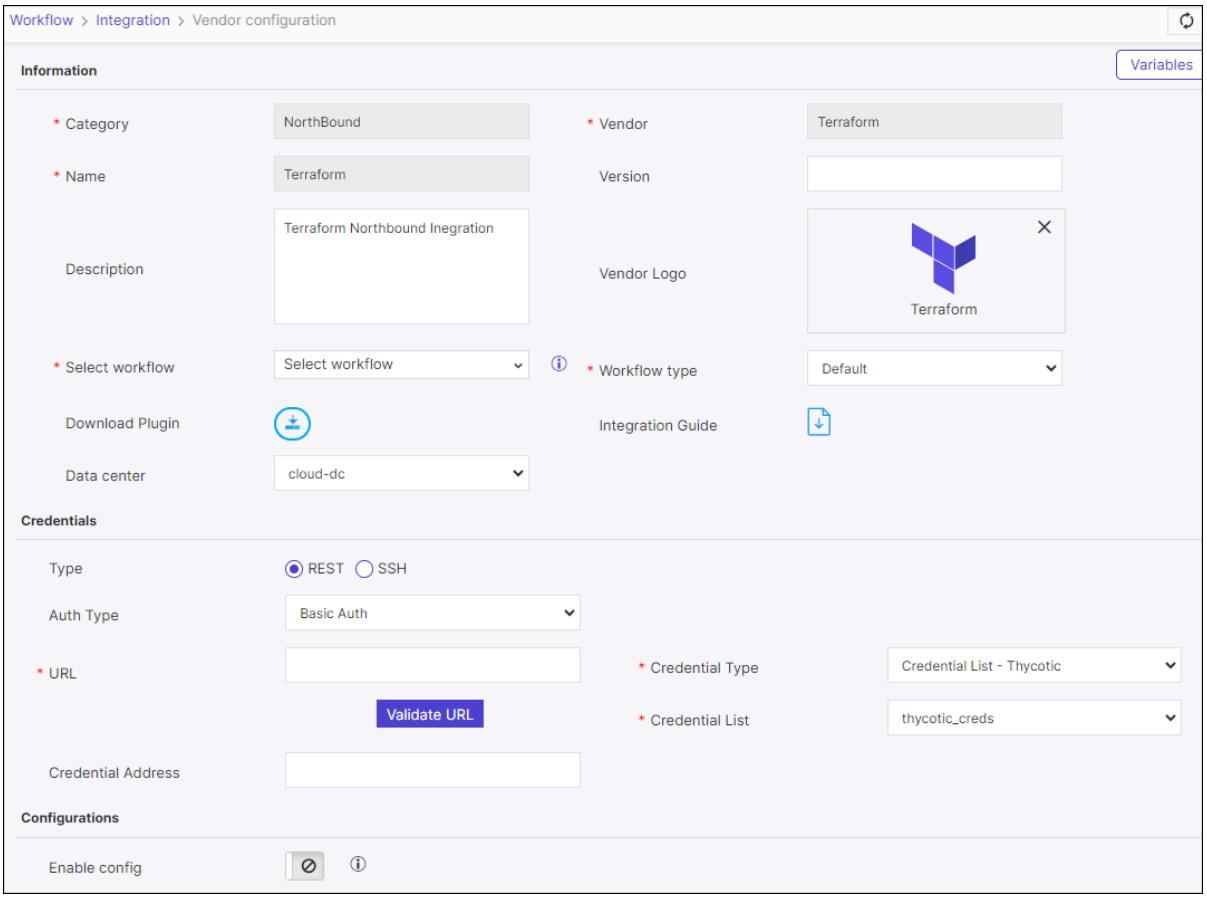
By integrating with third-party vendors such as Ansible, Bitbucket, Terraform and so on, you can define and use variables for your automation workflows.

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. Go to the Integration Hub.
2. On the **Workflow** > **Integration** page, click **Terraform**.
3. On the **Vendor Configuration** page, enter the field information to configure the Terraform integration.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: Under Authentication type **Basic Auth** and **Credentials** you can select the credential type from the dropdown and enter the relevant details to support credential authentication.

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>%>`

Vendor Configurations for PKI+

- Configuring Azure
- Configuring Cloudflare
- Configuring Infoblox

Configuring Azure

1. Go to **Menu** > **WORKFLOW** > **Integration**.
2. On the Integrations page, from the **Select Category** dropdown, select only **PKI** and click **Apply**.
3. Click the **Add new vendor** icon.
   The **Select Vendor** pop-up is displayed.
4. Select **Cloudflare**.
   The **Vendor Configuration** pop-up is displayed.
5. In the **Information** section, enter the following values:

Field descriptions for Information section

Field	Description
***Category polling**	Default value as *PKI*
***Vendor**	Default value as *Azure*
***Name**	Enter a suitable name.
Version	Enter a appropriate version.
Description	Enter a description for the vendor.
Upload image	Provision to upload vendor image/logo.
Tags	Select/Enter the tag.
Data center	Select data center value.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. In the **Credentials** section, based on the **Type** as ***REST*** and ***SSH***, enter the following values:
   If the **Type** = **REST**, the **Auth Type** has the following values in the dropdown - *Basic Auth*, *Credentials*, *Bearer Token*, *API*, *AWS Signature*, *Digest Auth*, *Akamai EdgeGrid*, *OAuth 2.0*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = REST, and Auth Type = Basic Auth or Credentials

Fields	Description
***Credential Type**	Select credential type as ***Manual*** or ***Credential List - XXX***. **Note:** The credentials must be added in the respective **Device Inventory**.
***Username**	This field is displayed if **Credential Type** = ***Manual***E Enter the username to authenticate and authorize communication with the vendor's system.
***Password**	This field is displayed if **Credential Type** = ***Manual*** Enter the password to authenticate and authorize communication with the vendor's system.
***Credential List**	This field is displayed if **Credential Type** = ***Credential List - XXX*** Select the configured device credentials.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Bearer Token

Fields	Description
***Token**	Enter the authentication token required for API requests.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
**: Mandatory fields*	

If Type = REST, and Auth Type = API Key

Fields	Description
***Key**	Enter the name of the API key parameter.
***Value**	Enter the actual API key used for authentication.
***Add to**	Select where the API key should be included in the request - *Header*, or *Query Parameter*.
Custom Header	If the API key needs to be sent in a specific header, enable the toggle and enter the key and value for defining a custom header name for it.
Custom Header Config	This field is displayed if the **Custom Header** toggle is enabled. Enter the *key* and *value* for defining a custom header name for it.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = AWS Signature

Fields	Description
***Access Key**	Enter the AWS access key ID associated with the AWS account, used for authentication.
***Secret Key**	Enter the secret key corresponding to the access key, used to sign API requests securely.
Advanced Configs	Enable the toggle button for additional configuration settings for customizing the authentication process.
***AWS Region**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the AWS region (e.g., `us-east-1`),where the API request should be sent

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***Service Name**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the AWS service being accessed (e.g., `s3`, `ec2`), required for signing the request correctly.
***Session Token**	This field is displayed if the **Advanced Configs** toggle is enabled. A temporary session token used when using AWS temporary security credentials (e.g., from AWS STS).
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Digest Auth

Fields	Description
***Username**	Enter the username used to authenticate with the vendor's API.
***Password**	Enter the corresponding password for the username.
Upload file	If authentication requires a certificate or key file, this field allows uploading the necessary credentials.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.
Realm	This field is displayed if the **Advanced Configs** toggle is enabled. A protection space defined by the server to indicate the scope of authentication (e.g., `"example.com"`).
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, one-time token generated by the server to prevent replay attacks.
***Algorithm**	This field is displayed if the **Advanced Configs** toggle is enabled. The hashing algorithm used for authentication (e.g., `MD5`, `SHA-256`).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
qop	This field is displayed if the **Advanced Configs** toggle is enabled. (Quality of Protection) Defines the security level of the authentication (e.g., `auth` for authentication only, `auth-int` for authentication with integrity protection).
Nonce Count	This field is displayed if the **Advanced Configs** toggle is enabled. A counter that tracks the number of requests made with the same nonce to prevent replay attacks.
Client Nonce	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, random value generated by the client to add randomness to the authentication process.
Opaque	This field is displayed if the **Advanced Configs** toggle is enabled. A server-specified string that must be returned unchanged by the client in subsequent requests.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Akamai EdgeGrid

Fields	Description
***Access Token**	Enter the token that uniquely identifies the client and grants access to Akamai's APIs.
***Client Secret**	Enter the secret key used to sign the request for secure authentication.
***Client Token**	Enter the token that identifies the client making the request and is required for authentication.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, one-time random string used to prevent replay attacks and enhance security.
***Timestamp**	This field is displayed if the **Advanced Configs** toggle is enabled. The time when the request is made, ensuring freshness and preventing request reuse.
***Base URL**	This field is displayed if the **Advanced Configs** toggle is enabled. The root URL of the Akamai API endpoint that requests will be sent to.
***Headers to sign**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the specific HTTP headers that should be included in the request signature to ensure integrity and security.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = OAuth 2.0

Fields	Description
***Username**	Used in **Grant Type = Password** , enter the username for authentication. This field is mandatory only if the Grant Type = Password.
***Password**	Used in **Grant Type = Password** , enter the password for the username. This field is mandatory only if the Grant Type = Password.
***Access Token URL**	Enter the provider's authentication server, to exchange an authorization code for an access token.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***Client ID**	Enter the ID for the client application registered with the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Client Secret	Enter the client secret given by the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Grant Type	Select the OAuth 2.0 flow to be used: • **Password** (for user authentication with username and password). • **Client Credentials** (for machine-to-machine authentication).
Scope	Defines the level of access requested (e.g., `read`, `write`, `admin`).
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If the **Type** = **SSH**, the **Auth Type** has the following values in the dropdown - *Credentials*, *SSH Key*, *No Auth*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = SSH, and Auth Type = Credentials

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP or FQDN/Host name**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	Enter the login username used to authenticate the SSH session.
***Password**	Enter the password for the corresponding username.
**: Mandatory fields*	

If Type = SSH, and Auth Type = SSH Key

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***SSH Key**	Enter the private SSH key used for authentication instead of a password.
**: Mandatory fields*	

If Type = SSH, and Auth Type = No Auth

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	The username used for the SSH session, though no authentication (password or key) is required.
**: Mandatory fields*	

7. Under **Configurations**, turn on the **Enable config** to add additional vendor configurations/attributes in JSON format. Attributes can be referred as variables.

Note: For a list of variables in workflow automation click the **Variables** button on the top right of the screen. A **Field variable mapping** pop-up is displayed with the label name and its corresponding variable name.

8. To save the changes, click **Add**. (Use the **Reset** button to clear all entered data and restores the fields to their default values.)

Configuring Cloudflare

1. Go to **Menu** > **WORKFLOW** > **Integration**.
2. On the Integrations page, from the **Select Category** dropdown, select only **PKI** and click **Apply**.
3. Click the **Add new vendor** icon.
   The **Select Vendor** pop-up is displayed.
4. Select **Cloudflare**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The **Vendor Configuration** pop-up is displayed.

5. In the **Information** section, enter the following values:

Field descriptions for Information section

Field	Description
***Category polling**	Default value as *PKI*
***Vendor**	Default value as *Cloudflare*
***Name**	Enter a suitable name.
Version	Enter a appropriate version.
Description	Enter a description for the vendor.
Upload image	Provision to upload vendor image/logo.
Tags	Select/Enter the tag.
Data center	Select data center value.

6. In the **Credentials** section, based on the **Type** as ***REST*** and ***SSH***, enter the following values:
If the **Type** = **REST**, the **Auth Type** has the following values in the dropdown - *Basic Auth*, *Credentials*, *Bearer Token*, *API*, *AWS Signature*, *Digest Auth*, *Akamai EdgeGrid*, *OAuth 2.0*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = REST, and Auth Type = Basic Auth or Credentials

Fields	Description
***Credential Type**	Select credential type as ***Manual*** or ***Credential List - XXX***. **Note:** The credentials must be added in the respective **Device Inventory**.
***Username**	This field is displayed if **Credential Type** = ***Manual***E Enter the username to authenticate and authorize communication with the vendor's system.
***Password**	This field is displayed if **Credential Type** = ***Manual***

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	Enter the password to authenticate and authorize communication with the vendor's system.
***Credential List**	This field is displayed if **Credential Type** = ***Credential List - XXX*** Select the configured device credentials.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Bearer Token

Fields	Description
***Token**	Enter the authentication token required for API requests.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = API Key

Fields	Description
***Key**	Enter the name of the API key parameter.
***Value**	Enter the actual API key used for authentication.
***Add to**	Select where the API key should be included in the request - *Header*, or *Query Parameter*.
Custom Header	If the API key needs to be sent in a specific header, enable the toggle and enter the key and value for defining a custom header name for it.
Custom Header Config	This field is displayed if the **Custom Header** toggle is enabled. Enter the *key* and *value* for defining a custom header name for it.
***URL**	Enter the valid URL/endpoint of the vendor’s API.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = AWS Signature

Fields	Description
***Access Key**	Enter the AWS access key ID associated with the AWS account, used for authentication.
***Secret Key**	Enter the secret key corresponding to the access key, used to sign API requests securely.
Advanced Configs	Enable the toggle button for additional configuration settings for customizing the authentication process.
***AWS Region**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the AWS region (e.g., `us-east-1`),where the API request should be sent
***Service Name**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the AWS service being accessed (e.g., `s3`, `ec2`), required for signing the request correctly.
***Session Token**	This field is displayed if the **Advanced Configs** toggle is enabled. A temporary session token used when using AWS temporary security credentials (e.g., from AWS STS).
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Digest Auth

Fields	Description
***Username**	Enter the username used to authenticate with the vendor's API.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***Password**	Enter the corresponding password for the username.
Upload file	If authentication requires a certificate or key file, this field allows uploading the necessary credentials.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.
Realm	This field is displayed if the **Advanced Configs** toggle is enabled. A protection space defined by the server to indicate the scope of authentication (e.g., `"example.com"`).
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, one-time token generated by the server to prevent replay attacks.
***Algorithm**	This field is displayed if the **Advanced Configs** toggle is enabled. The hashing algorithm used for authentication (e.g., `MD5`, `SHA-256`).
qop	This field is displayed if the **Advanced Configs** toggle is enabled. (Quality of Protection) Defines the security level of the authentication (e.g., `auth` for authentication only, `auth-int` for authentication with integrity protection).
Nonce Count	This field is displayed if the **Advanced Configs** toggle is enabled. A counter that tracks the number of requests made with the same nonce to prevent replay attacks.
Client Nonce	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, random value generated by the client to add randomness to the authentication process.
Opaque	This field is displayed if the **Advanced Configs** toggle is enabled. A server-specified string that must be returned unchanged by the client in subsequent requests.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Akamai EdgeGrid

Fields	Description
***Access Token**	Enter the token that uniquely identifies the client and grants access to Akamai's APIs.
***Client Secret**	Enter the secret key used to sign the request for secure authentication.
***Client Token**	Enter the token that identifies the client making the request and is required for authentication.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, one-time random string used to prevent replay attacks and enhance security.
***Timestamp**	This field is displayed if the **Advanced Configs** toggle is enabled. The time when the request is made, ensuring freshness and preventing request reuse.
***Base URL**	This field is displayed if the **Advanced Configs** toggle is enabled. The root URL of the Akamai API endpoint that requests will be sent to.
***Headers to sign**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the specific HTTP headers that should be included in the request signature to ensure integrity and security.
***URL**	Enter the valid URL/endpoint of the vendor's API.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = OAuth 2.0

Fields	Description
***Username**	Used in **Grant Type = Password** , enter the username for authentication. This field is mandatory only if the Grant Type = Password.
***Password**	Used in **Grant Type = Password** , enter the password for the username. This field is mandatory only if the Grant Type = Password.
***Access Token URL**	Enter the provider's authentication server, to exchange an authorization code for an access token.
***Client ID**	Enter the ID for the client application registered with the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Client Secret	Enter the client secret given by the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Grant Type	Select the OAuth 2.0 flow to be used: • **Password** (for user authentication with username and password). • **Client Credentials** (for machine-to-machine authentication).
Scope	Defines the level of access requested (e.g., `read`, `write`, `admin`).
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

If the **Type** = **SSH**, the **Auth Type** has the following values in the dropdown - *Credentials*, *SSH Key*, *No Auth*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = SSH, and Auth Type = Credentials

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP or FQDN/Host name**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	Enter the login username used to authenticate the SSH session.
***Password**	Enter the password for the corresponding username.
**: Mandatory fields*	

If Type = SSH, and Auth Type = SSH Key

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***SSH Key**	Enter the private SSH key used for authentication instead of a password.
**: Mandatory fields*	

If Type = SSH, and Auth Type = No Auth

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	The username used for the SSH session, though no authentication (password or key) is required.
**: Mandatory fields*	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. Under **Configurations**, turn on the **Enable config** to add additional vendor configurations/attributes in JSON format. Attributes can be referred as variables.

   Note: For a list of variables in workflow automation click the **Variables** button on the top right of the screen. A **Field variable mapping** pop-up is displayed with the label name and its corresponding variable name.

8. To save the changes, click **Add**. (Use the **Reset** button to clear all entered data and restores the fields to their default values.)

Configuring Infoblox

1. Go to **Menu** > **WORKFLOW** > **Integration**.
2. On the Integrations page, from the **Select Category** dropdown, select only **PKI** and click **Apply**.
3. Click the **Add new vendor** icon.
   The **Select Vendor** pop-up is displayed.
4. Select **Cloudflare**.
   The **Vendor Configuration** pop-up is displayed.
5. In the **Information** section, enter the following values:

Field descriptions for Information section

Field	Description
***Category polling**	Default value as *PKI*
***Vendor**	Default value as *Infoblox*
***Name**	Enter a suitable name.
Version	Enter a appropriate version.
Description	Enter a description for the vendor.
Upload image	Provision to upload vendor image/logo.
Tags	Select/Enter the tag.
Data center	Select data center value.

6. In the **Credentials** section, based on the **Type** as ***REST*** and ***SSH***, enter the following values:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

If the **Type** = **REST**, the **Auth Type** has the following values in the dropdown - *Basic Auth*, *Credentials*, *Bearer Token*, *API*, *AWS Signature*, *Digest Auth*, *Akamai EdgeGrid*, *OAuth 2.0*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = REST, and Auth Type = Basic Auth or Credentials

Fields	Description
***Credential Type**	Select credential type as ***Manual*** or ***Credential List - XXX***. **Note:** The credentials must be added in the respective **Device Inventory**.
***Username**	This field is displayed if **Credential Type** = ***Manual***E Enter the username to authenticate and authorize communication with the vendor's system.
***Password**	This field is displayed if **Credential Type** = ***Manual*** Enter the password to authenticate and authorize communication with the vendor's system.
***Credential List**	This field is displayed if **Credential Type** = ***Credential List - XXX*** Select the configured device credentials.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Bearer Token

Fields	Description
***Token**	Enter the authentication token required for API requests.
***URL**	Enter the valid URL/endpoint of the vendor’s API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

If Type = REST, and Auth Type = API Key

Fields	Description
***Key**	Enter the name of the API key parameter.
***Value**	Enter the actual API key used for authentication.
***Add to**	Select where the API key should be included in the request - *Header*, or *Query Parameter*.
Custom Header	If the API key needs to be sent in a specific header, enable the toggle and enter the key and value for defining a custom header name for it.
Custom Header Config	This field is displayed if the **Custom Header** toggle is enabled. Enter the *key* and *value* for defining a custom header name for it.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = AWS Signature

Fields	Description
***Access Key**	Enter the AWS access key ID associated with the AWS account, used for authentication.
***Secret Key**	Enter the secret key corresponding to the access key, used to sign API requests securely.
Advanced Configs	Enable the toggle button for additional configuration settings for customizing the authentication process.
***AWS Region**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the AWS region (e.g., `us-east-1`),where the API request should be sent
***Service Name**	This field is displayed if the **Advanced Configs** toggle is enabled.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	Enter the AWS service being accessed (e.g., `s3`, `ec2`), required for signing the request correctly.
***Session Token**	This field is displayed if the **Advanced Configs** toggle is enabled. A temporary session token used when using AWS temporary security credentials (e.g., from AWS STS).
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Digest Auth

Fields	Description
***Username**	Enter the username used to authenticate with the vendor's API.
***Password**	Enter the corresponding password for the username.
Upload file	If authentication requires a certificate or key file, this field allows uploading the necessary credentials.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.
Realm	This field is displayed if the **Advanced Configs** toggle is enabled. A protection space defined by the server to indicate the scope of authentication (e.g., `"example.com"`).
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, one-time token generated by the server to prevent replay attacks.
***Algorithm**	This field is displayed if the **Advanced Configs** toggle is enabled. The hashing algorithm used for authentication (e.g., `MD5`, `SHA-256`).
qop	This field is displayed if the **Advanced Configs** toggle is enabled.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	(Quality of Protection) Defines the security level of the authentication (e.g., `auth` for authentication only, `auth-int` for authentication with integrity protection).
Nonce Count	This field is displayed if the **Advanced Configs** toggle is enabled. A counter that tracks the number of requests made with the same nonce to prevent replay attacks.
Client Nonce	This field is displayed if the **Advanced Configs** toggle is enabled. A unique, random value generated by the client to add randomness to the authentication process.
Opaque	This field is displayed if the **Advanced Configs** toggle is enabled. A server-specified string that must be returned unchanged by the client in subsequent requests.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = Akamai EdgeGrid

Fields	Description
***Access Token**	Enter the token that uniquely identifies the client and grants access to Akamai's APIs.
***Client Secret**	Enter the secret key used to sign the request for secure authentication.
***Client Token**	Enter the token that identifies the client making the request and is required for authentication.
Advanced Configs	Enable the toggle button for additional configurations for customizing the authentication request.
***Nonce**	This field is displayed if the **Advanced Configs** toggle is enabled.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	A unique, one-time random string used to prevent replay attacks and enhance security.
***Timestamp**	This field is displayed if the **Advanced Configs** toggle is enabled. The time when the request is made, ensuring freshness and preventing request reuse.
***Base URL**	This field is displayed if the **Advanced Configs** toggle is enabled. The root URL of the Akamai API endpoint that requests will be sent to.
***Headers to sign**	This field is displayed if the **Advanced Configs** toggle is enabled. Enter the specific HTTP headers that should be included in the request signature to ensure integrity and security.
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If Type = REST, and Auth Type = OAuth 2.0

Fields	Description
***Username**	Used in **Grant Type = Password** , enter the username for authentication. This field is mandatory only if the Grant Type = Password.
***Password**	Used in **Grant Type = Password** , enter the password for the username. This field is mandatory only if the Grant Type = Password.
***Access Token URL**	Enter the provider's authentication server, to exchange an authorization code for an access token.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***Client ID**	Enter the ID for the client application registered with the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Client Secret	Enter the client secret given by the API provider. This field is mandatory only if the Grant Type = Client Credentials.
Grant Type	Select the OAuth 2.0 flow to be used: • **Password** (for user authentication with username and password). • **Client Credentials** (for machine-to-machine authentication).
Scope	Defines the level of access requested (e.g., `read`, `write`, `admin`).
***URL**	Enter the valid URL/endpoint of the vendor's API. To validate the URL Click **Validate URL**.
**: Mandatory fields*	

If the **Type** = **SSH**, the **Auth Type** has the following values in the dropdown - *Credentials*, *SSH Key*, *No Auth*. The table below has the corresponding field descriptions for each of the Auth Types.

If Type = SSH, and Auth Type = Credentials

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP or FQDN/Host name**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	Enter the login username used to authenticate the SSH session.
***Password**	Enter the password for the corresponding username.
**: Mandatory fields*	

If Type = SSH, and Auth Type = SSH Key

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***SSH Key**	Enter the private SSH key used for authentication instead of a password.
**: Mandatory fields*	

If Type = SSH, and Auth Type = No Auth

Fields	Description
Communication type	Select the communication type as ***IP*** or ***FQDN/Host name***
***IP**	Enter the IP address or the FQDN/Host name of the vendor's server that AppViewX will connect to over SSH.
***Port**	Enter the SSH port on the vendor's server.
***Username**	The username used for the SSH session, though no authentication (password or key) is required.
**: Mandatory fields*	

7. Under **Configurations**, turn on the **Enable config** to add additional vendor configurations/attributes in JSON format. Attributes can be referred as variables.

> **Note:** For a list of variables in workflow automation click the **Variables** button on the top right of the screen. A **Field variable mapping** pop-up is displayed with the label name and its corresponding variable name.

8. To save the changes, click **Add**. (Use the **Reset** button to clear all entered data and restores the fields to their default values.)

Workflow Tasks

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.

Tasks are broadly divided into the following categories:

- General
- User Interface
- ChatOps and Notifications
- Change Management.

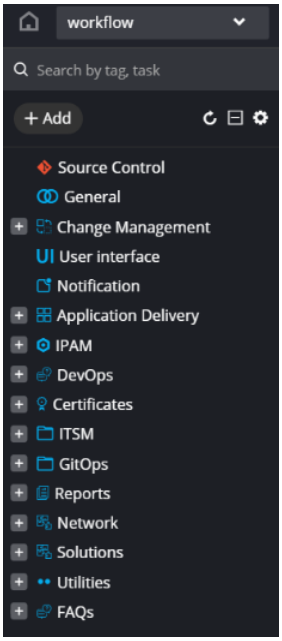
- Accessing Workflow Tasks
- Task Category - General
- Task Category - User Interface
- Task Category - ChatOps and Notifications
- Task Category - Change Management
- Provisioning for Workflow Tasks
- Connecting Workflow Tasks
- Adding Variables to Tasks
- Automation Collision
- Workflow Task Actions
- Rollback Workflow
- Variable Mapping
- Scheduling Tasks
- Workflow Cart
- Version Control
- Workflow Options
- Northbound Integrations

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Accessing Workflow Tasks

1. Go to the Workflow page.
2. From the **Workflow** page, click **Create New**.

   The design space is displayed with the workflow tasks listed in the menu on the left.



Options and Descriptions for Workflow Tasks folder menu

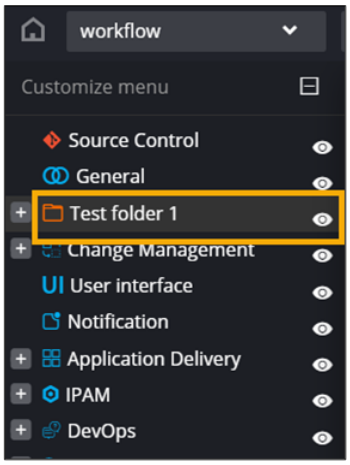
Options	Description
Search	Allows you to search for tasks by typing keyword(s)
Add	Allows you to add a new folder
	Refreshes the tasks folders menu
	Allows you to expand and collapse the task folders
	Allows you to hide folders from the workflow tasks menu and customize task sequence within a folder

Hiding a Task Folder

To hide a folder from the workflow tasks menu:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. From the top right corner of the menu, click **(Settings)** icon.
2. Click **(View)** icon next to the folder you want to hide from the menu.



3. Click **Save**.

   The folder will not be visible in the menu.

Reordering Task Sequence

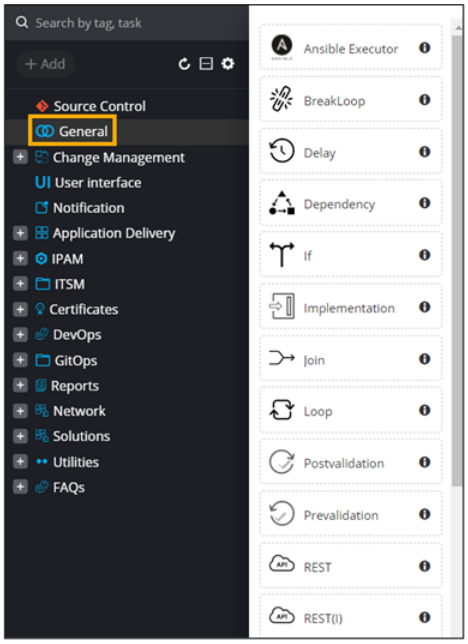
To reorder the sequence of tasks within a folder:

1. From the top right corner of the menu, click **(Settings)** icon.
2. Select a folder from the menu.
3. Hold and drag tasks to change their sequence in the folder.

Task Category - General

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- Ansible
- Break Loop
- Delay
- Dependency
- If
- Implementation
- Join
- Loop
- Postvaildation
- Prevalidation
- REST
- REST(I)
- Retry
- Rollback
- Schedule
- Script
- Split

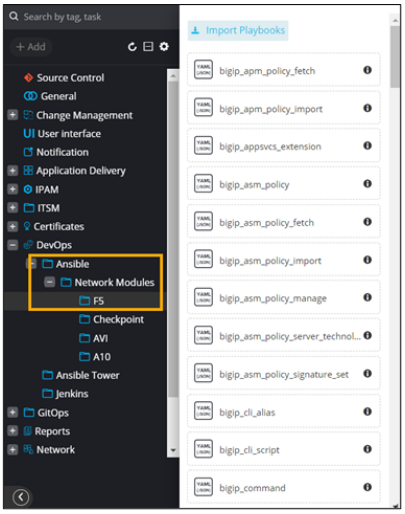
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Switch
- WorkOrder

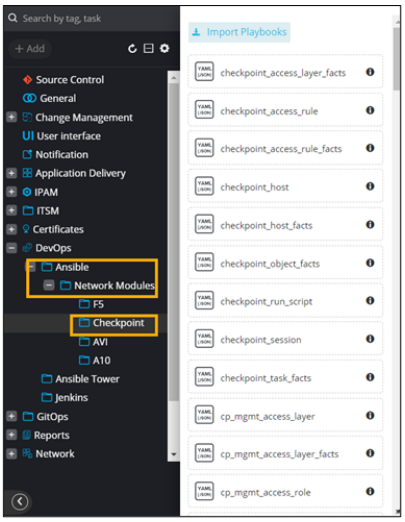
Ansible

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

- **F5**

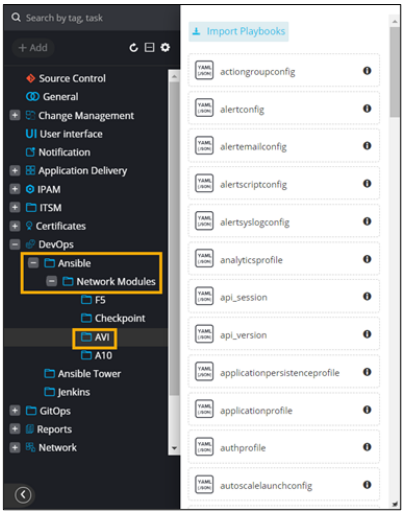


- **Checkpoint**

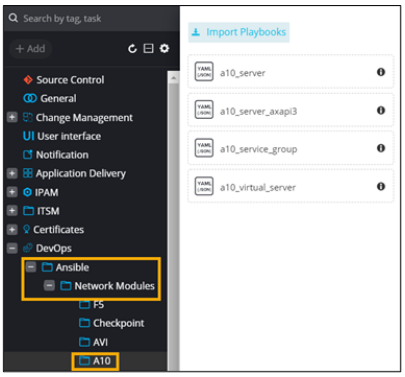


- **AVI**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- **A10**



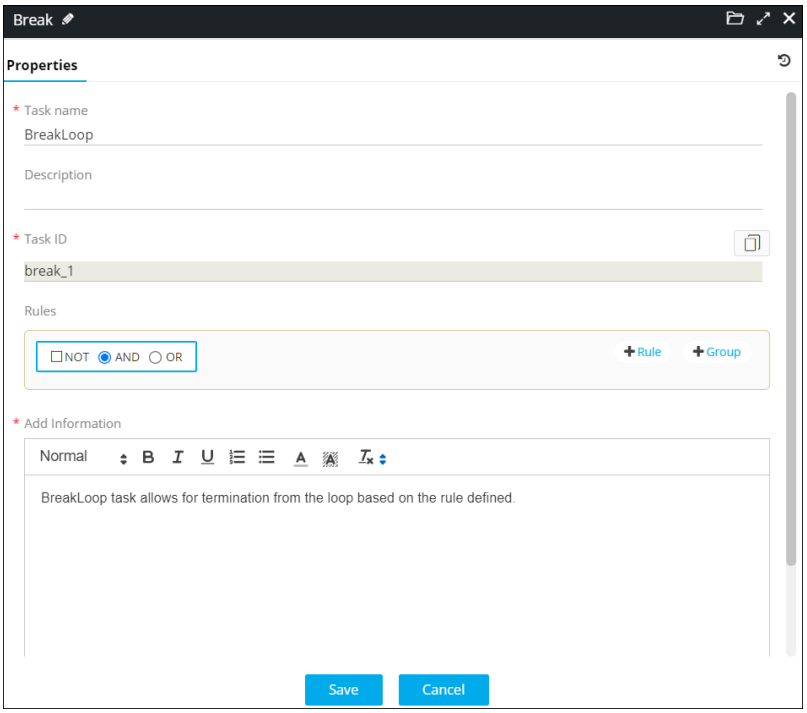
There is also a provision to import playbooks from Ansible.

Break Loop

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

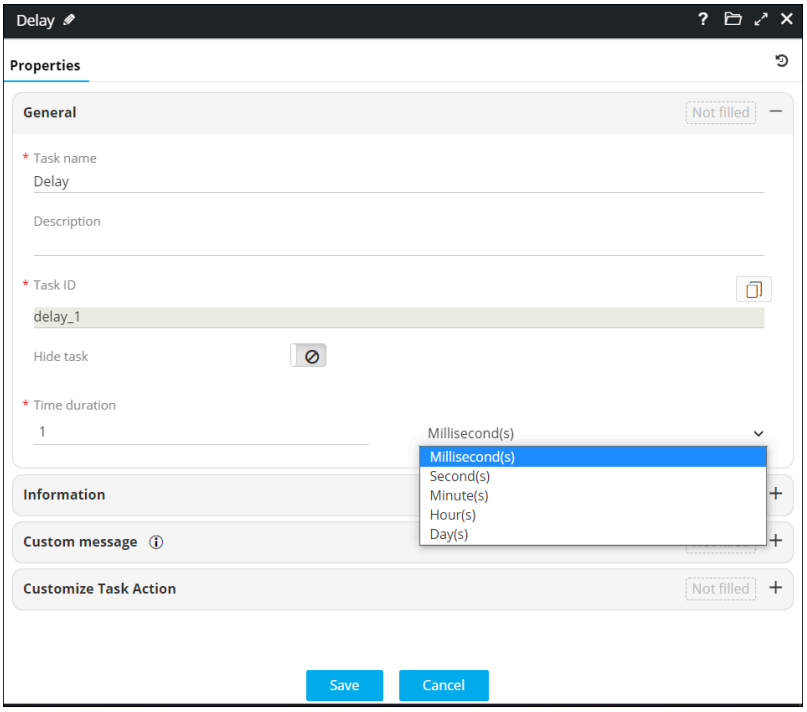
- Provision to define a break condition to terminate the loop.
- Provision to define rule or boolean condition (AND, OR, NOT) to test the loop variable.
  - If the condition holds true, the loop terminates.
  - If the condition holds false, control is passed to the loop.

Copyright © 2025 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 © 2025 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.

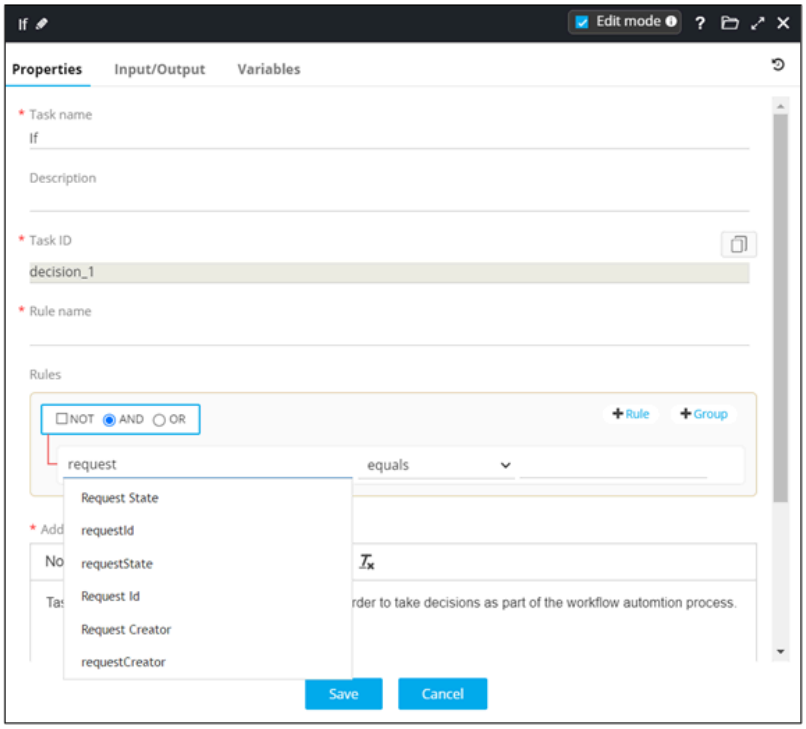
- Provision to select the dependent tasks which are defined within the workflow.
- Provision to assign a wait time or the duration for which the next task in the flow can wait for the dependent task to be completed.
- Task dependency will check for the successful completion of the dependent tasks.

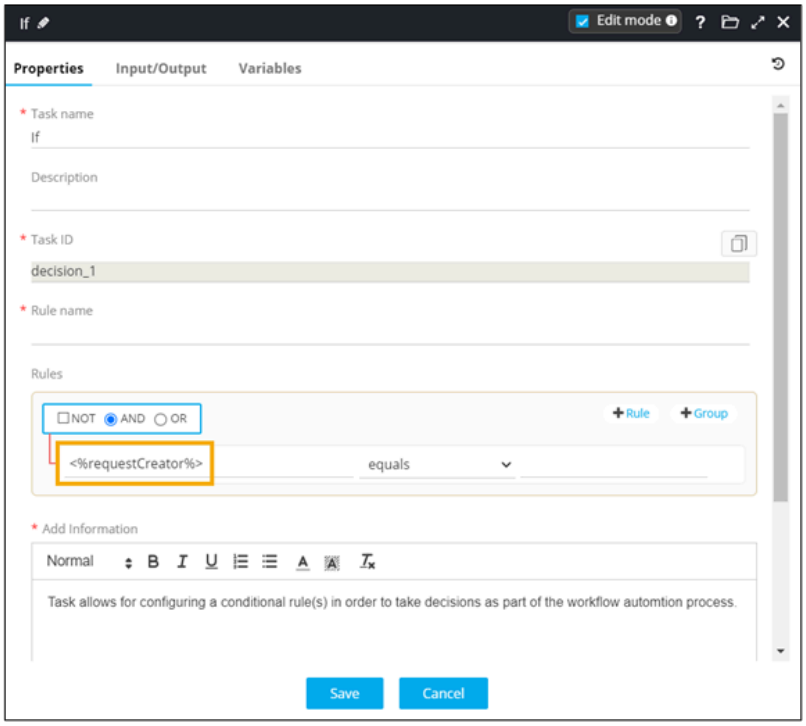
If

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 refer 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 © 2025 AppViewX, Inc. All Rights Reserved.

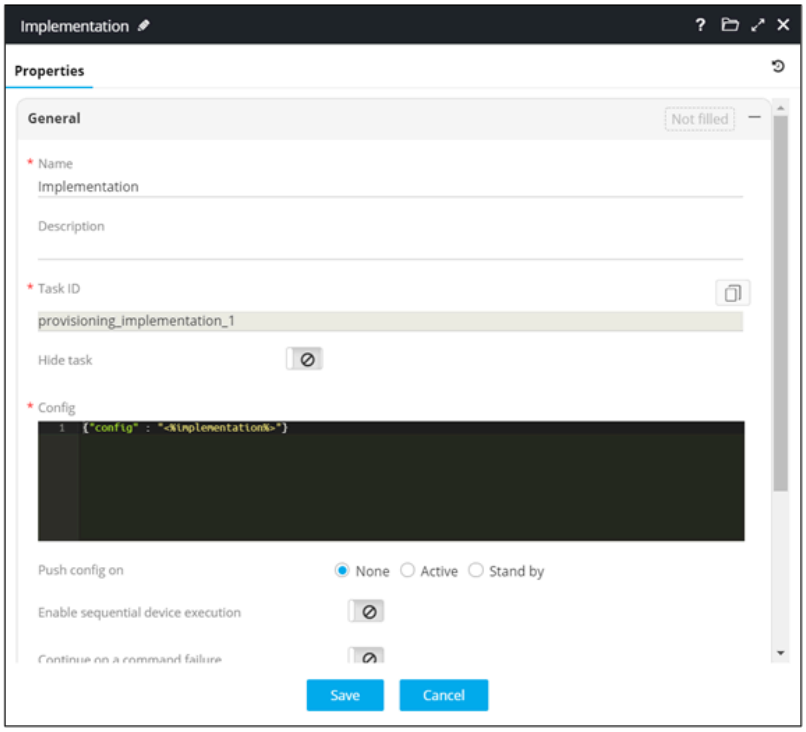




Implementation

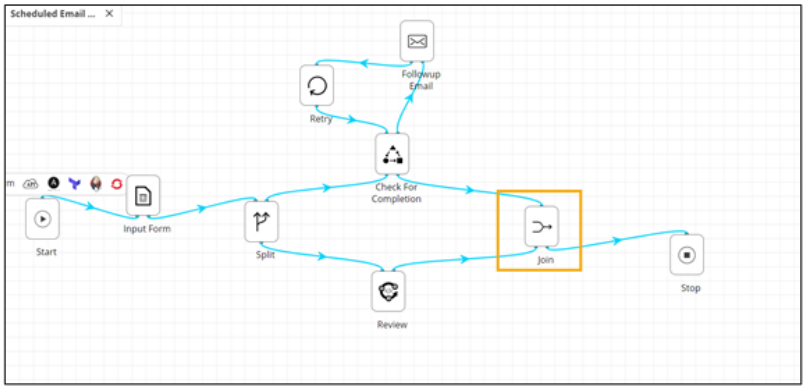
This task allows you to perform Implementation changes on the end point.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Join

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

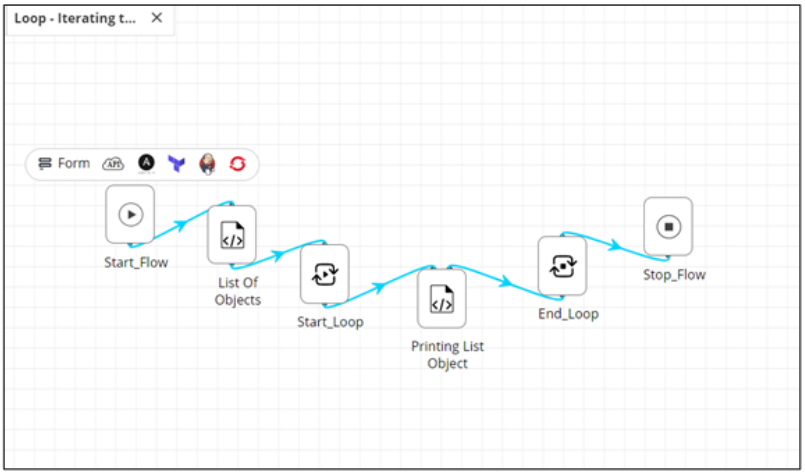


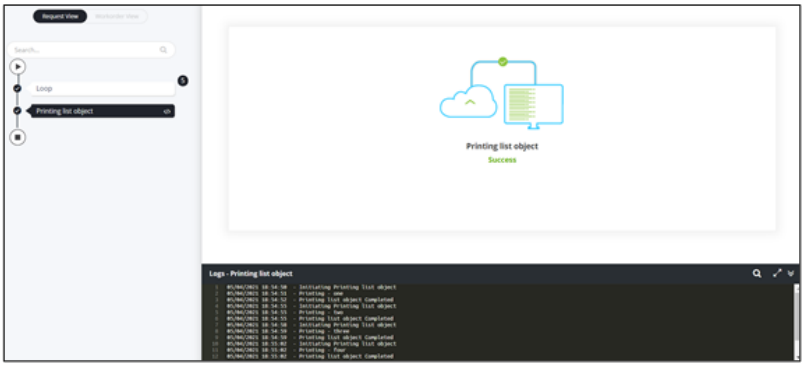
Loop

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

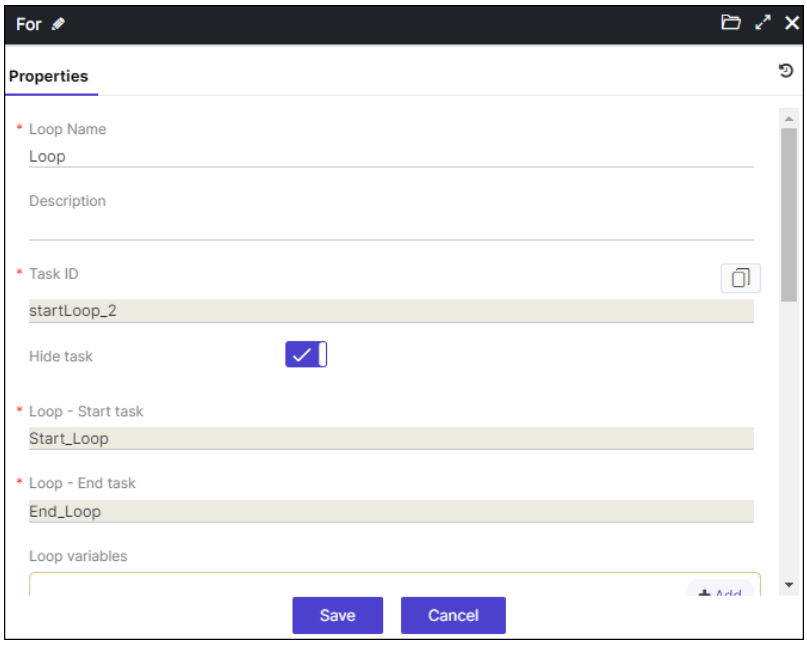
- Provision to define a loop for iterative execution of a task or sub-process.
- Provision to resume control to the next iteration of the loop in the event of failure of any task within the loop.
- Provision to define rule or Boolean condition (AND, OR, NOT) to execute the loop variable. If the condition 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.

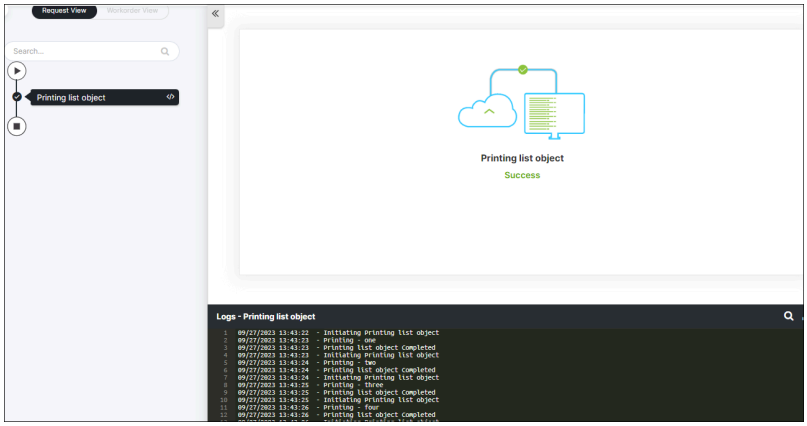




- Provision to enable or disable the visibility of loops in the stage view.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

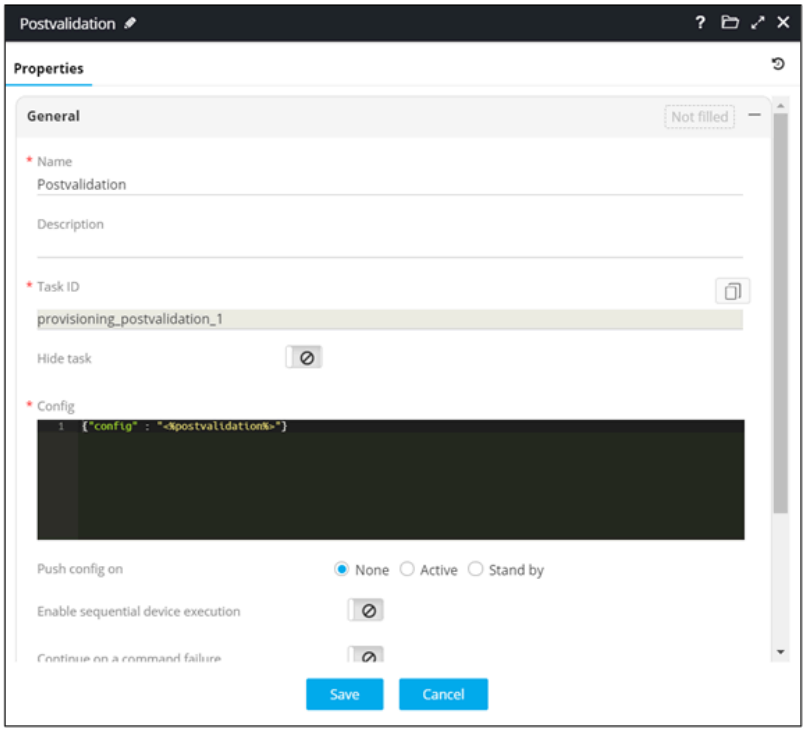




Postvaildation

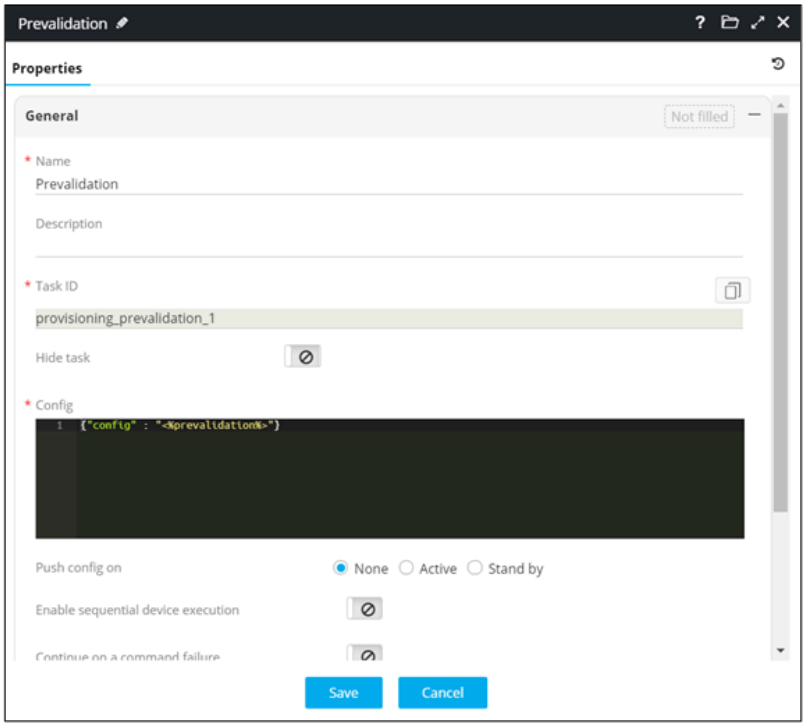
This task allows you to perform post-validation changes on the end point.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Prevalidation

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

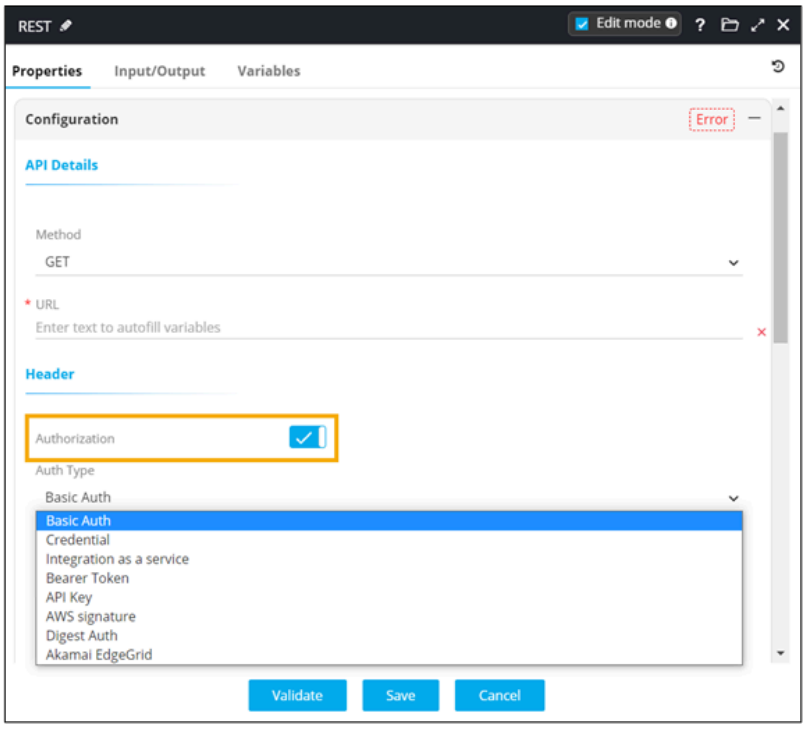


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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



- Authorization Modes

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Authorization Modes

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

- Basic Auth
- Credential
- Integration as a Service
- Bearer Token
- API Key
- AWS Signature
- Digest Auth
- Akamai EdgeGrid
- oAuth 2.0

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, enble the **Authorization**, select **Auth Type** as **Basic Auth**.

2. If the Credential Type is **Manual**, then Enter the **Username** and **Password**. (If you choose any preconfigured Credential Type from the dropdown, the select the appropriate **Credential List**.

> **Note:** The Password field is masked.

3. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Credential

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 **Credential**.
2. If the Credential Type is **Manual**, then Enter the **Username** and **Password**. (If you choose any preconfigured Credential Type from the dropdown, the select the appropriate **Credential List**.

> **Note:** The Password field is masked.

3. Click **Save**.

Integration as a Service

1. In the **REST** task, under **Authorization**, select **Auth Type** as **Integration as a Service**.
2. From the **Integration as a service** dropdown, select the type of service, such as GitHub, JIRA, BitBucket, etc.
3. Click **Save**.

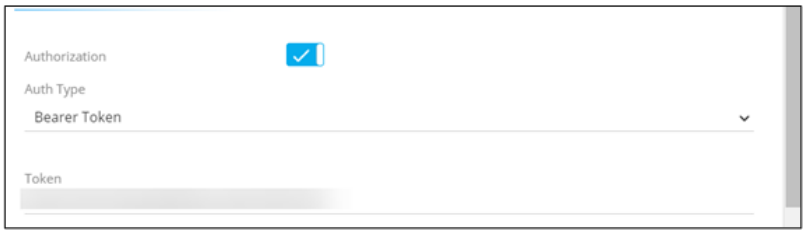
Bearer Token

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**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

2. Enter the **Token** value.



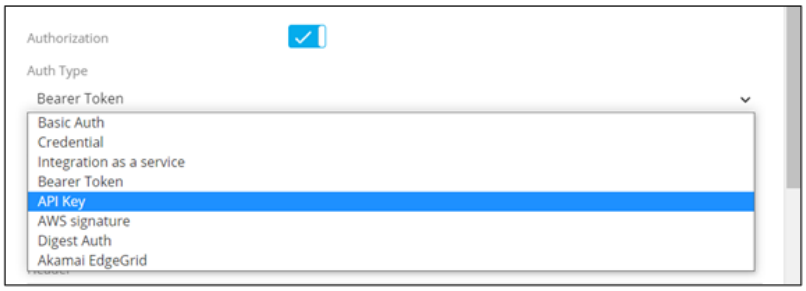
Note: The Token field is masked.

3. Click **Save**.

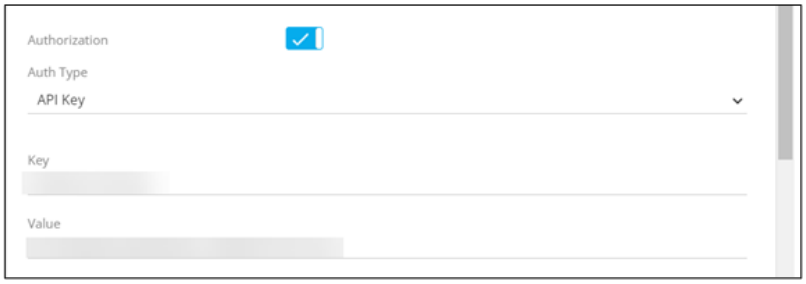
API Key

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.

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



2. Enter the **Key** and **Value**.



Note: The Value field is masked.

3. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

AWS Signature

Authentication information you send must include a signature.

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

2. Enter the **Access Key** and **Secret Key** details.

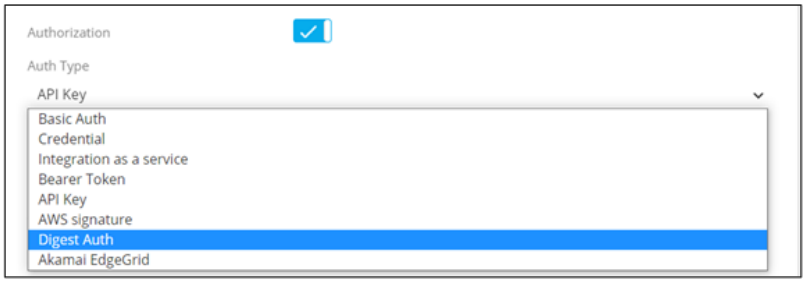
> **Note:** The **Access Key** and **Secret Key** fields are masked.

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.

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



2. Enter the **Username** and **Password**.

> **Note:** The Password field is masked.

3. Click **Browse** to upload the file.

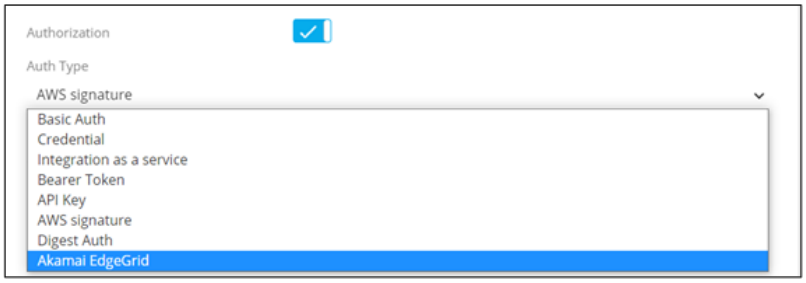
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. If the **Advanced** toggle is enabled then enter details in the following fields.
   - Realm
   - Nonce
   - Algorithm
   - qop
   - Nonce Count
   - Client Nonce
   - Opaque
5. 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**.



2. Enter the **Access Token**, **Client Secret**, and **Client Token**.

> **Note:** The **Access Token**, **Client Secret**, and **Client Token** fields are masked.

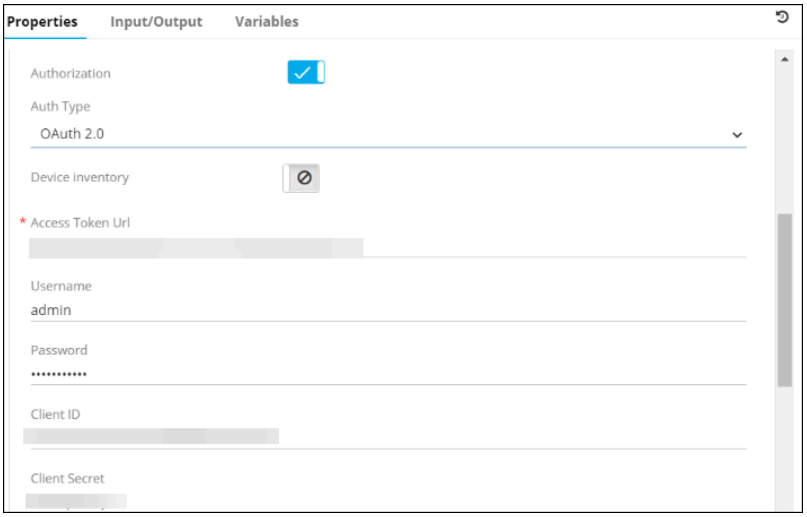
3. If the **Advanced** toggle is enabled then enter details in the following fields.
   - Nonce
   - Timestamp
   - Base URL
   - Headers to sign
4. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

oAuth 2.0

This authentication uses an access token that works as a temporary password (with fixed validity) instead of a regular password.

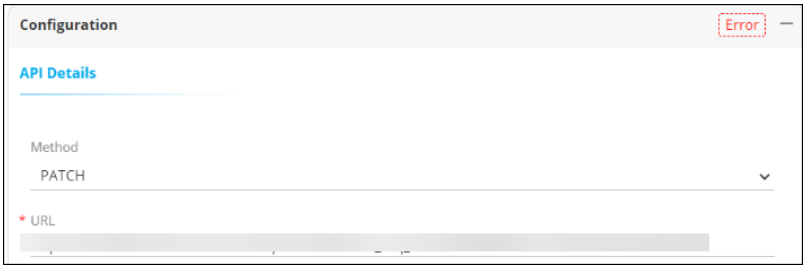
1. In the **REST** task, under **Authorization**, select **Auth Type** as **OAuth 2.0**.
2. Enter all the required field information.
   - Access Token URL of the oAuth server.
   - Username
   - Password
   - Client ID
   - Client Secret.



Note: The **Password** and **Client Secret** fields are masked.

Once this information is submitted, the client will receive an authorization code/refresh token.

3. Add this token received from the oAuth server to the **URL** in the **API Details** section.



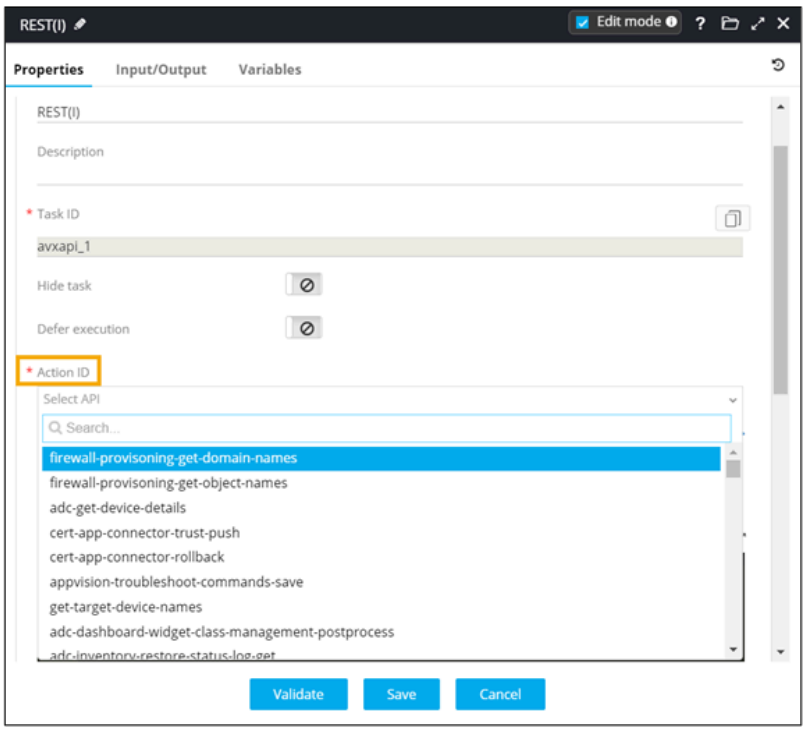
4. Click **Save**.

Copyright © 2025 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, Accounts and so on. 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.



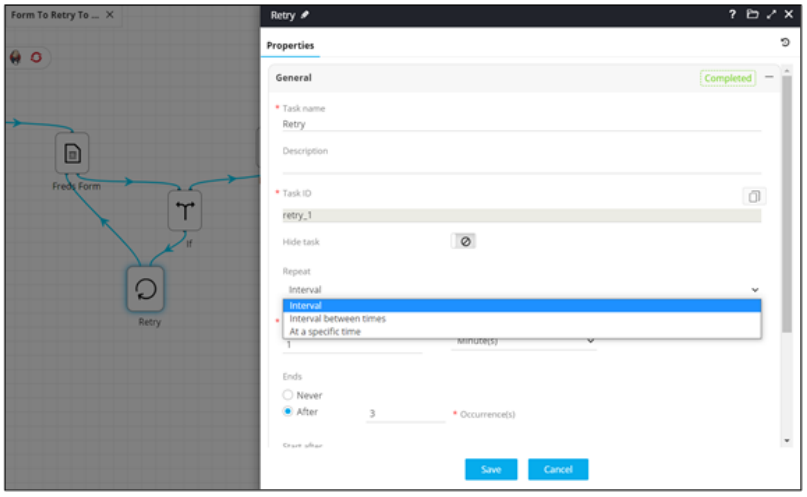
Note: Relevant APIs have to be added into the database collection (visualworkflow_avxapi) in order to be consumed as Action ID via the REST (I) task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

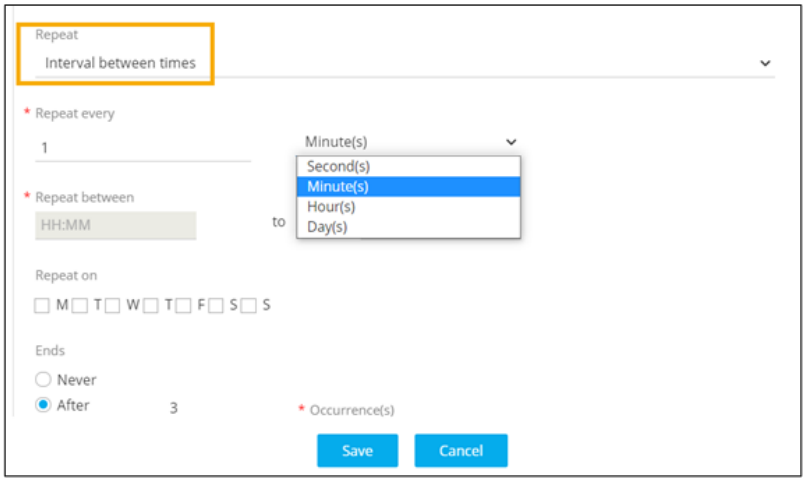
Retry

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

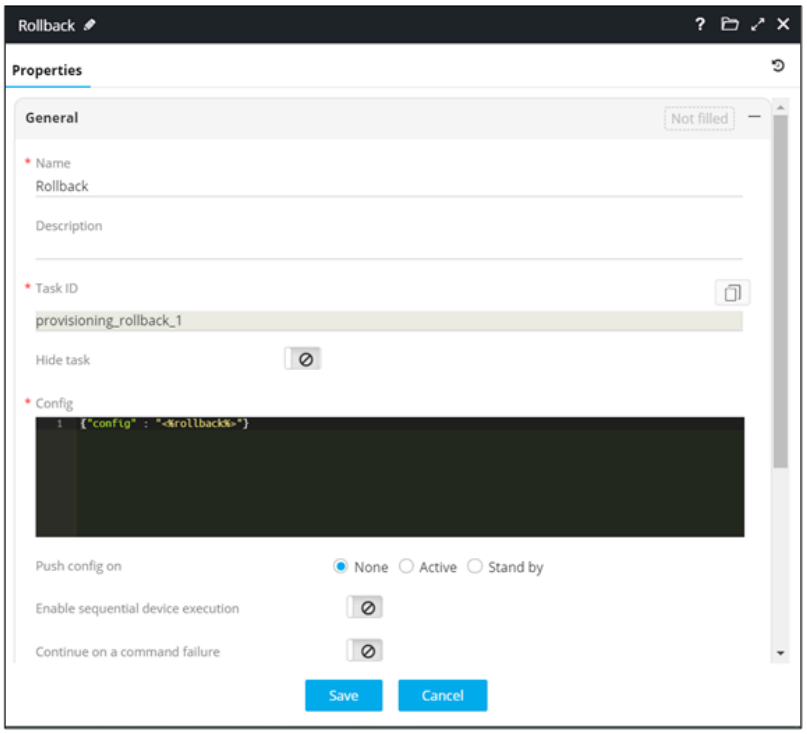


• **At a specific time**: Provision to retry only at a specific time, day before exit.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Rollback

This task allows you to rollback implemented changes.



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

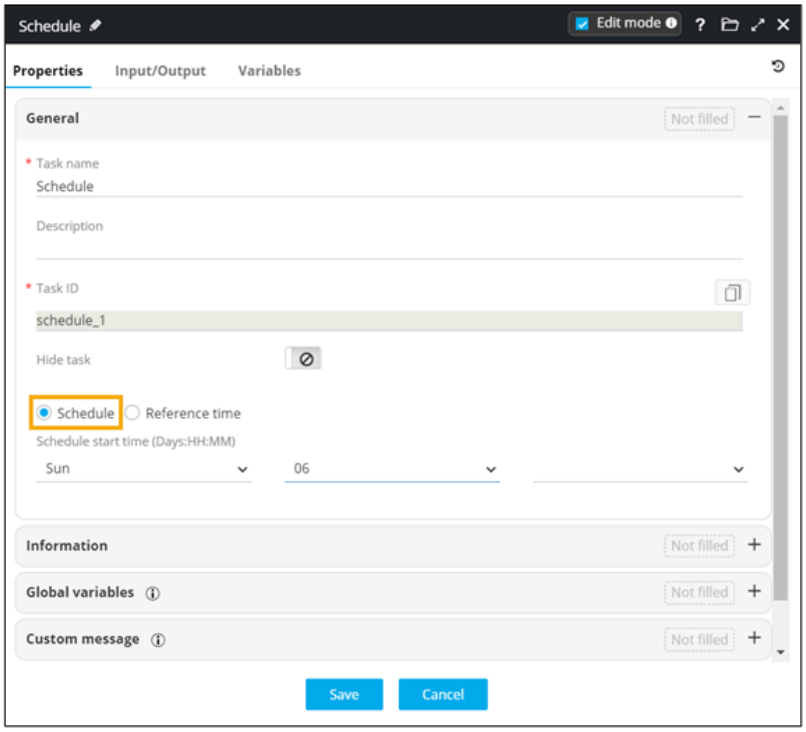
Schedule

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.



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

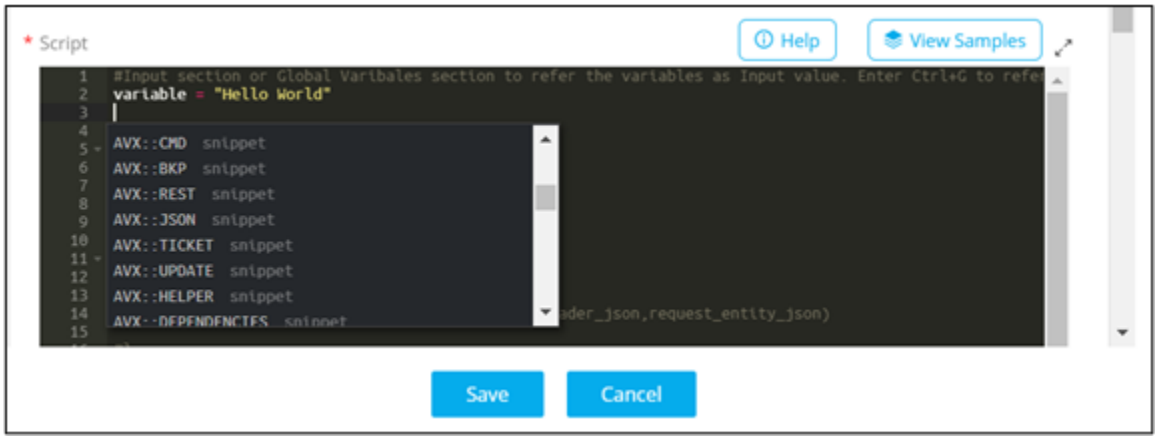
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.

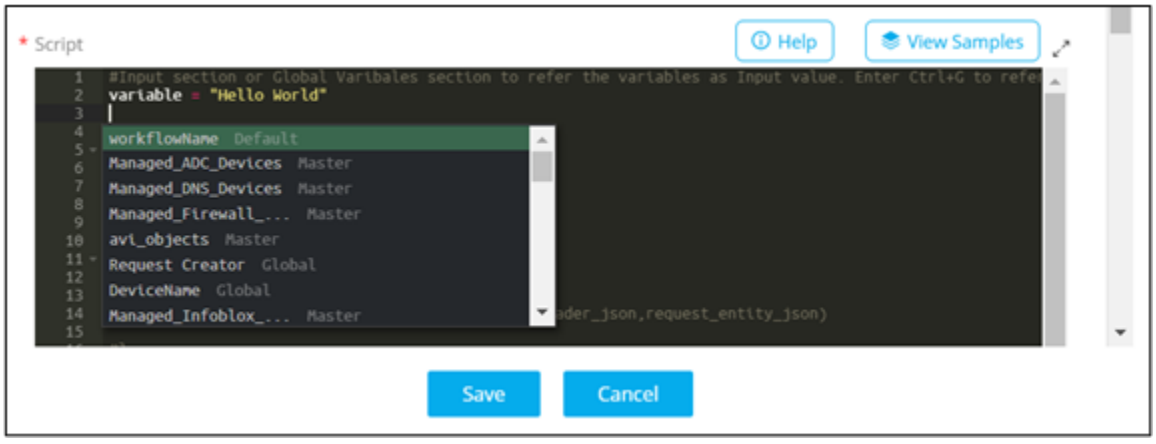
- Using Keyboard Shortcuts
- Custom Commands

Using Keyboard Shortcuts

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

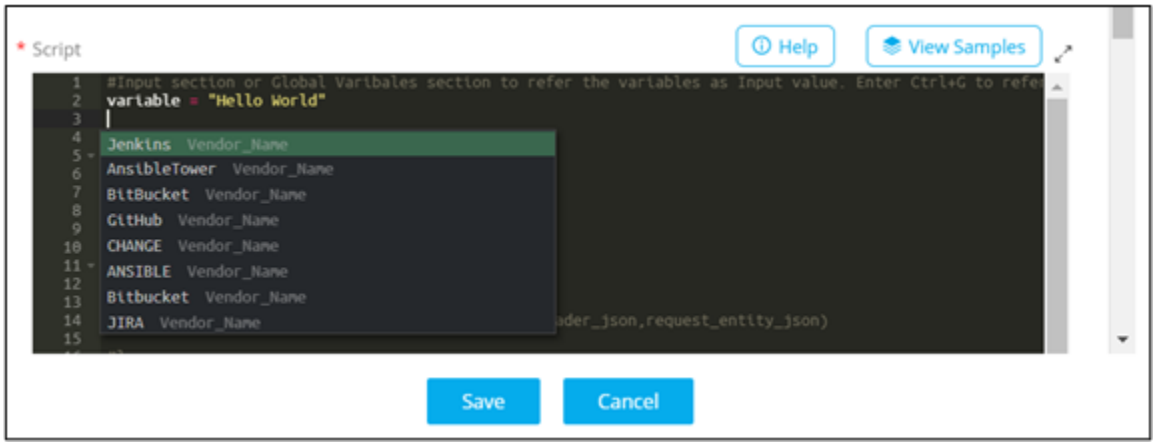


- Provision to use the keyboard shortcut **Ctrl+G** to refer global variable(s).



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Custom Commands

The 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 Workflow Studio:

- 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::ESCAPE

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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 referred anywhere across the workflow.

Usage:

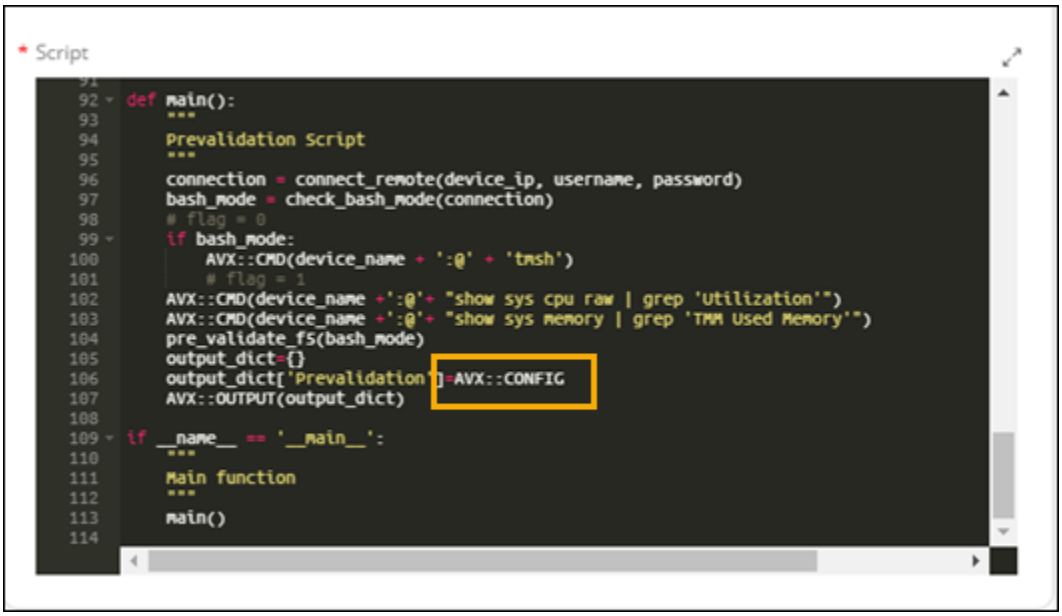
AVX::CMD(“testdevice”+':@'+”implementation command 1”) AVX::CMD(“testdevice”+':@'+”implementation command 2”) commands ={} commands[“implementation”] = AVX::CONFIG	• Commands 1 and 2 will get added to the implementation key. • This will add up the commands generated in the script to the provided key.

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 © 2025 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%>'
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
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 = []

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
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
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")
    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)
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    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)
    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)
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    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)
            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
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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

    output_dict['Rollback']=AVX::CONFIG

    AVX::OUTPUT(output_dict)

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

AVX::PUSH and AVX::JSONCONVERTER

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

Usage:

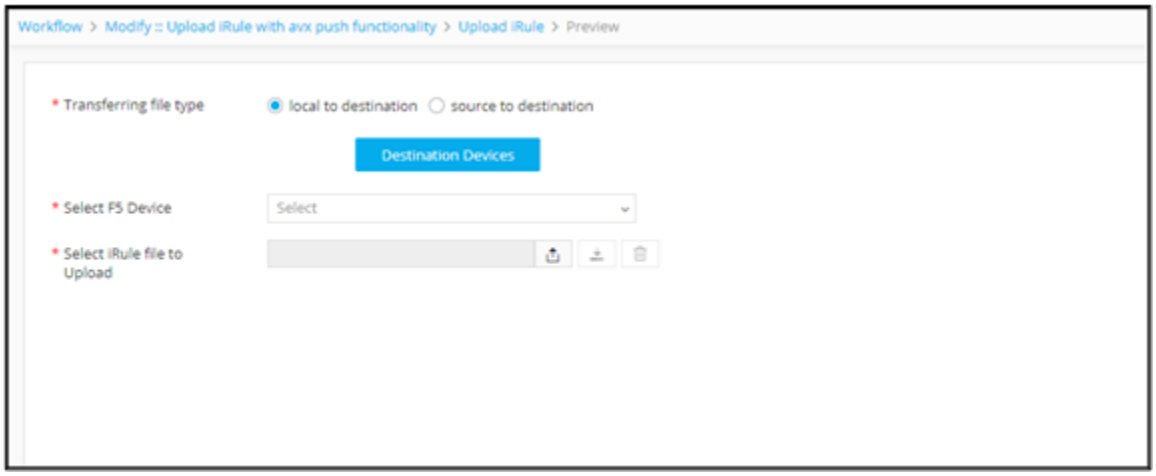
```
AVX::PUSH ('device_name',source_json, destination_json,protocol, fieldId , port_number, recursive, seq_no=0, properties={'sleep':0})
```

- 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

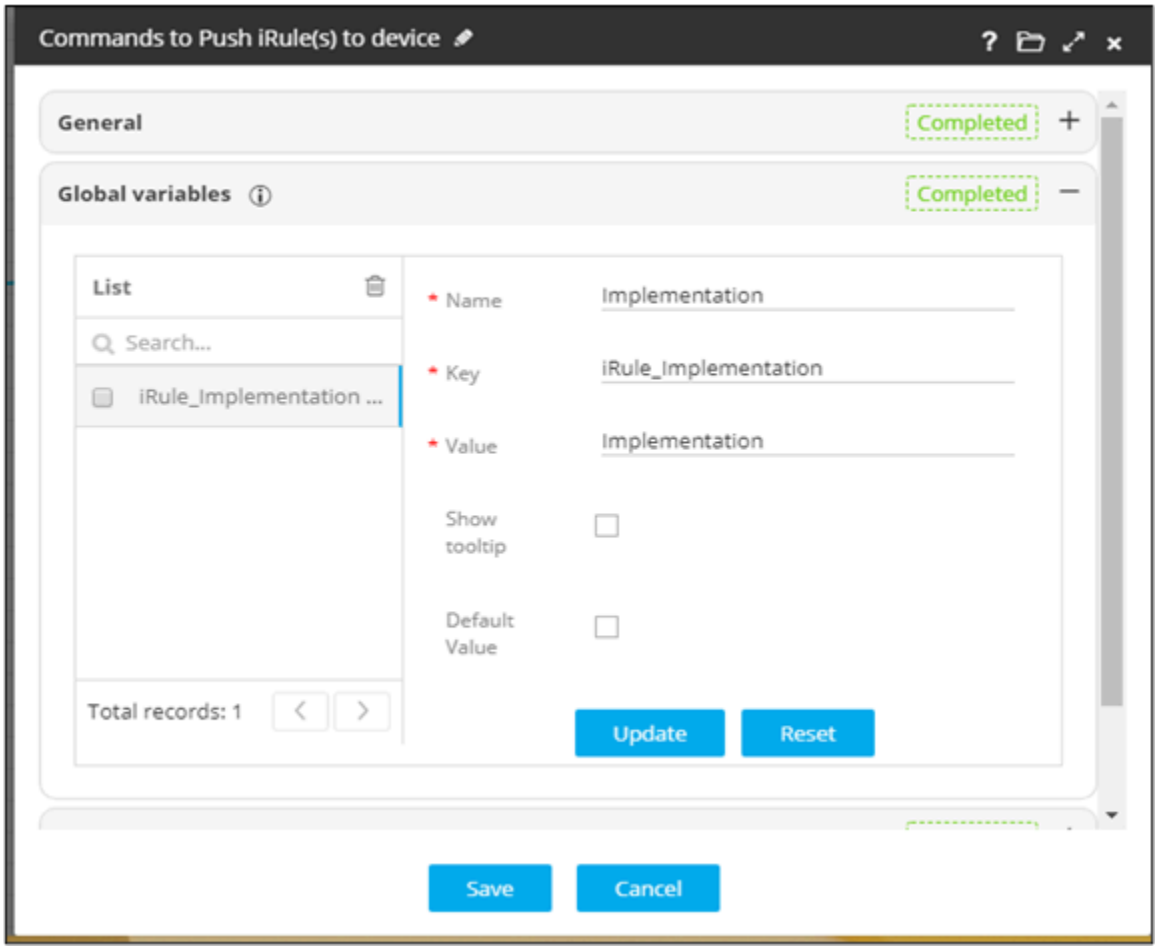
```
#load sys config file <irulepath> verify merge


import sys

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)
```

Copyright © 2025 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+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')
        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)
```

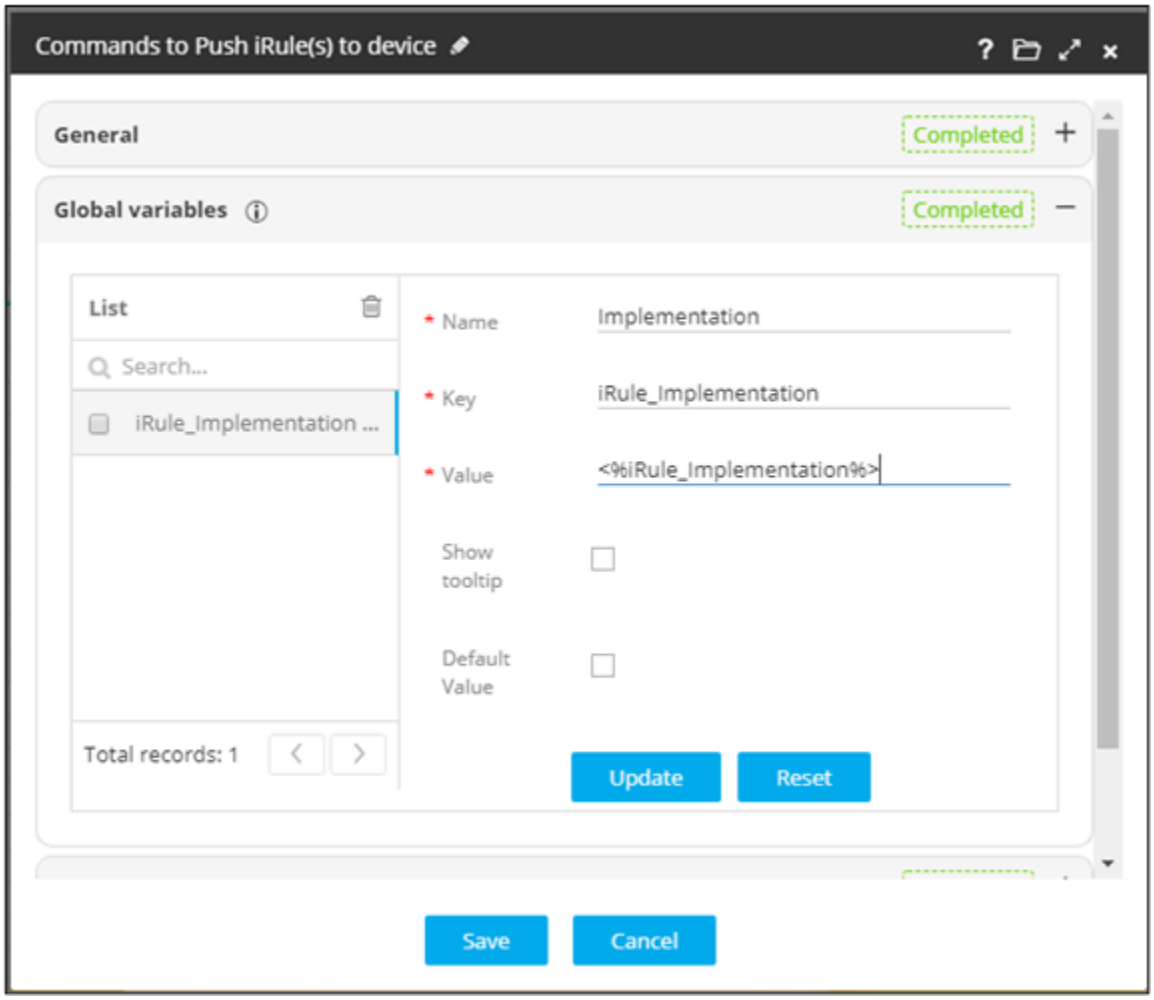
5. Define the configuration to be generated from the script as a global variable value.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



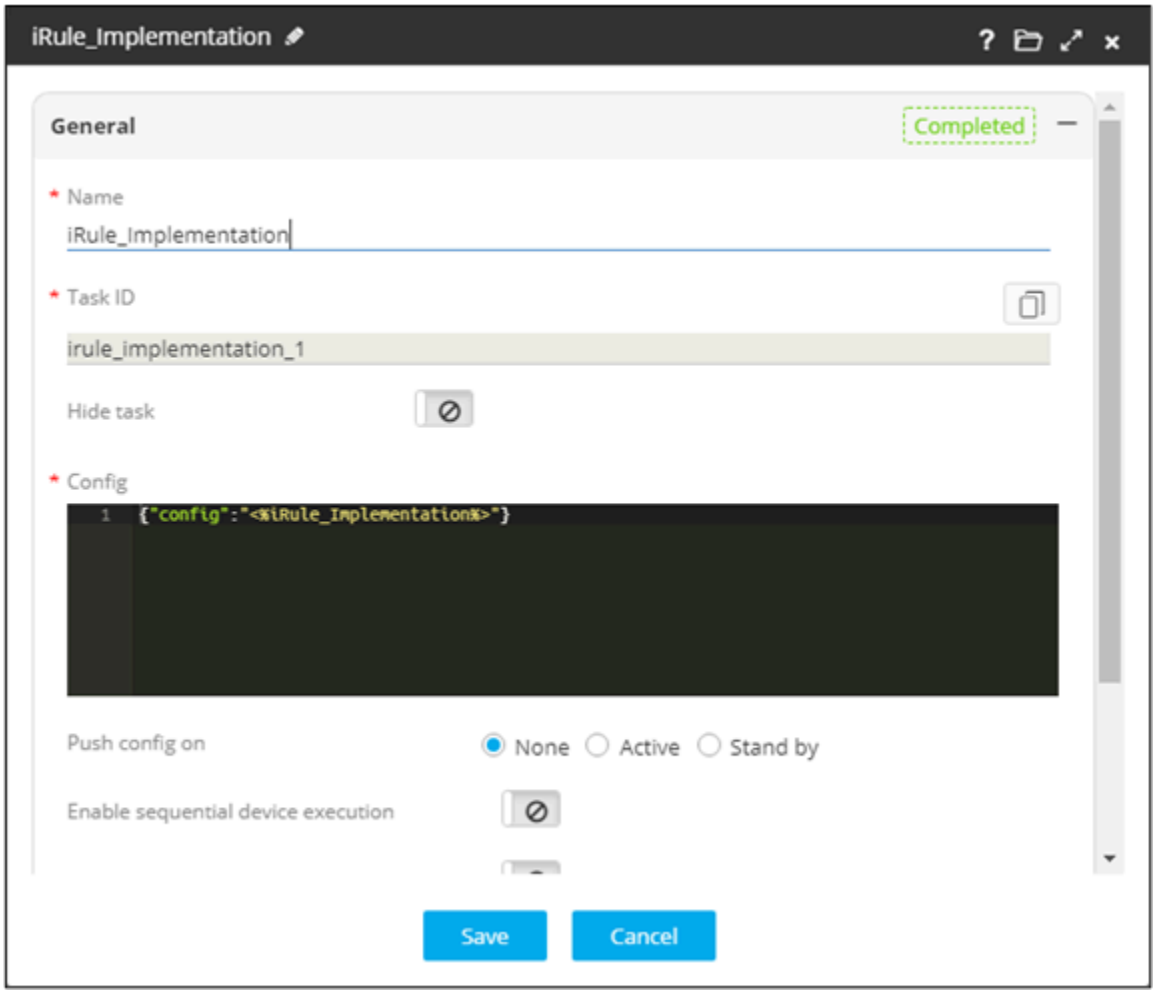
6. To display the configurations to be pushed to a device, drag and drop a **Review** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

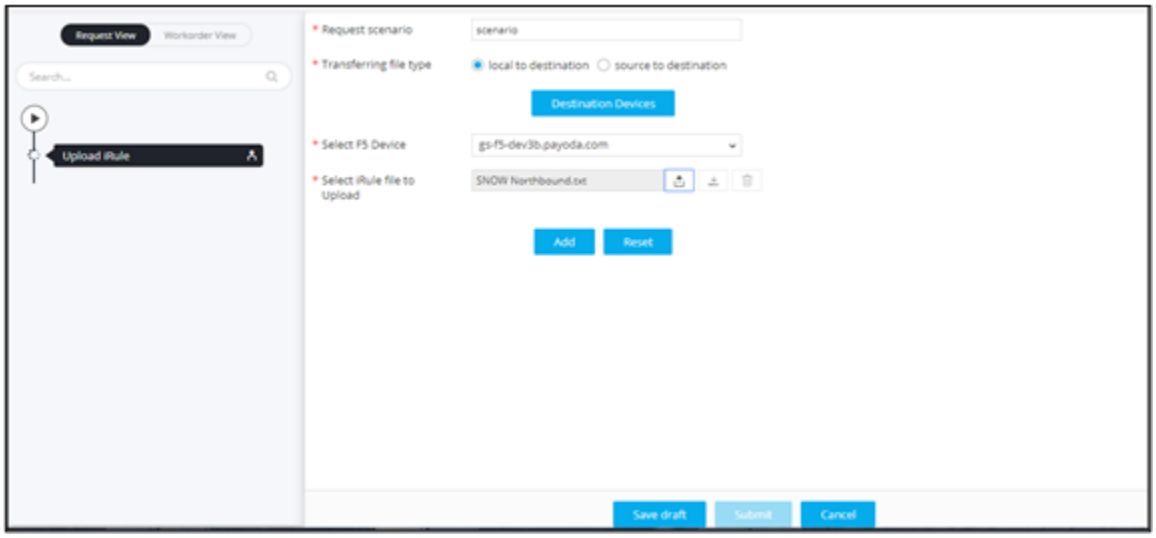
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



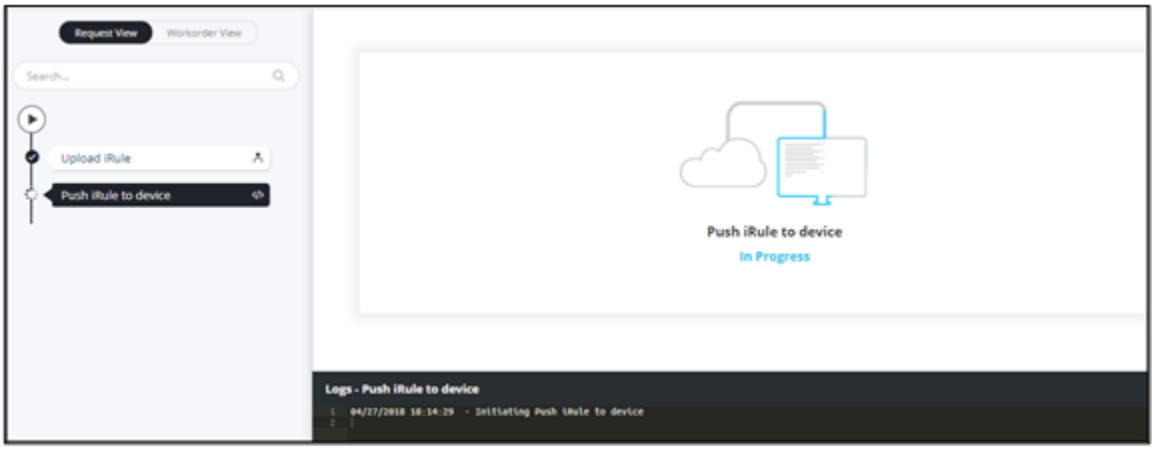
8. Refer the global variable from the Script task.
9. Enable and trigger the workflow.

   The workflow is executed with the user inputs requested at the first stage.
10. Select the destination device and upload the file.

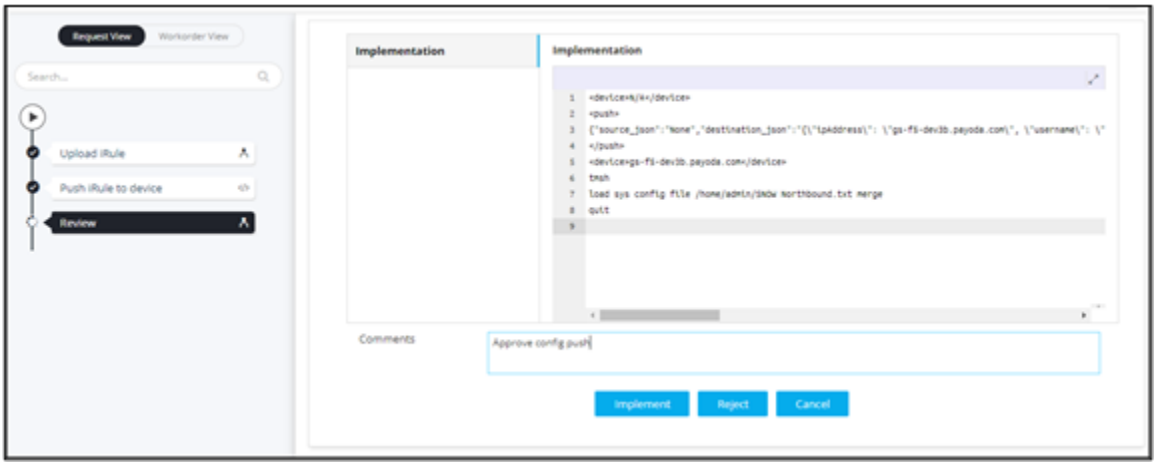
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- Push iRule to device task is in progress.

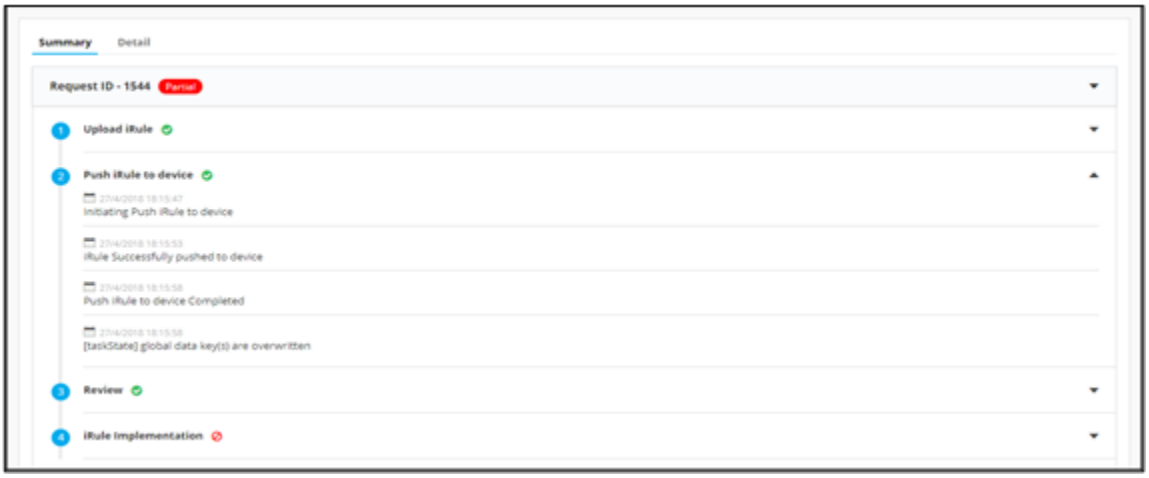


- Implementation stage.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.

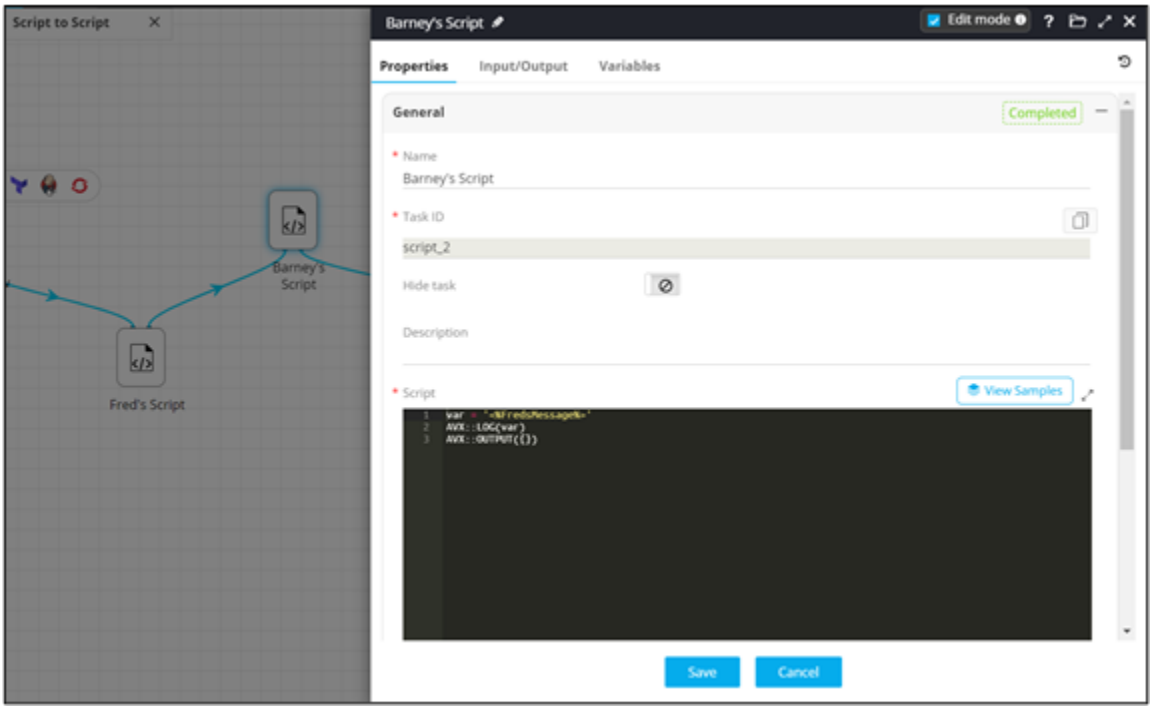
Usage:

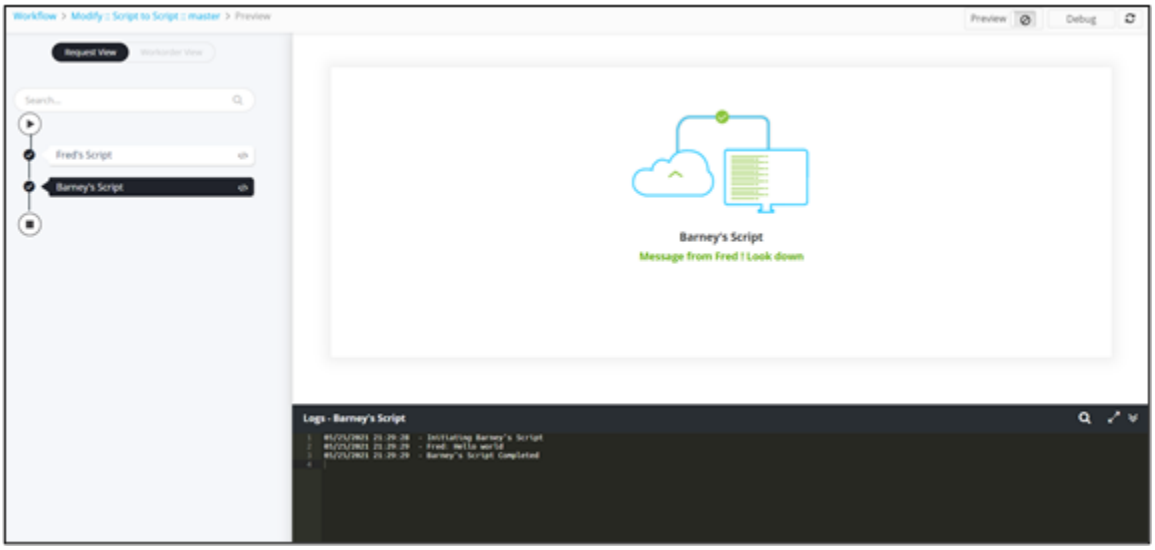
```
AVX::LOG(“Print a simple log message”)

AVX::OUTPUT({})
```

Example 1:

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

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

AVX::OUTPUT({})
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

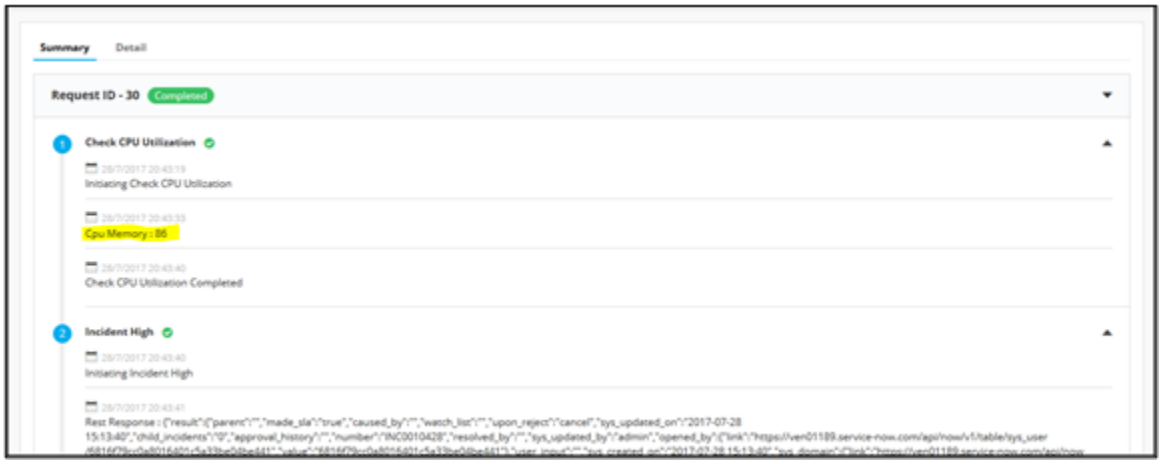




Example 2:

```
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))
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



AVX::CMD

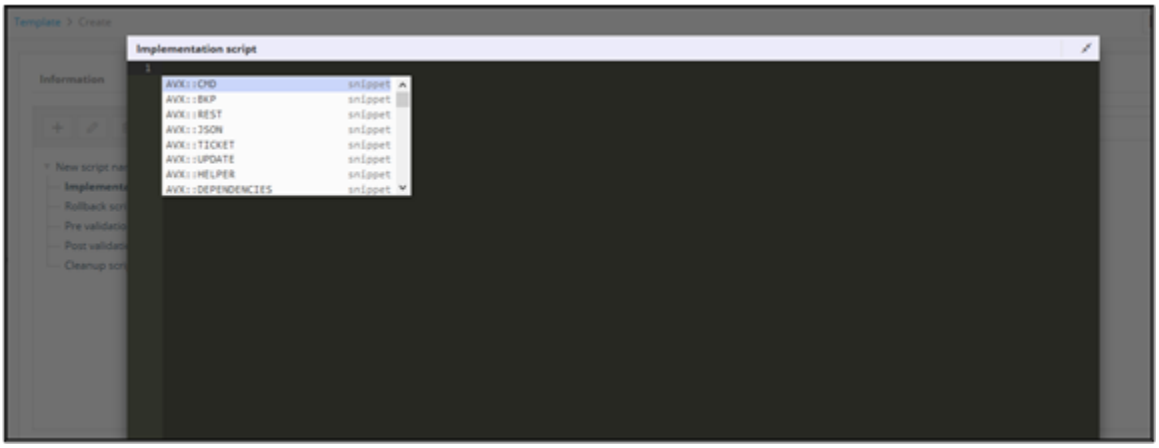
```
AVX::CMD('device_name' +':@'+ 'command',interactive='None|command|input',seq_no=0,properties={'sleep':0,'wait':0})
```

String	Description
device name	Name of the device on which the command needs to be executed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

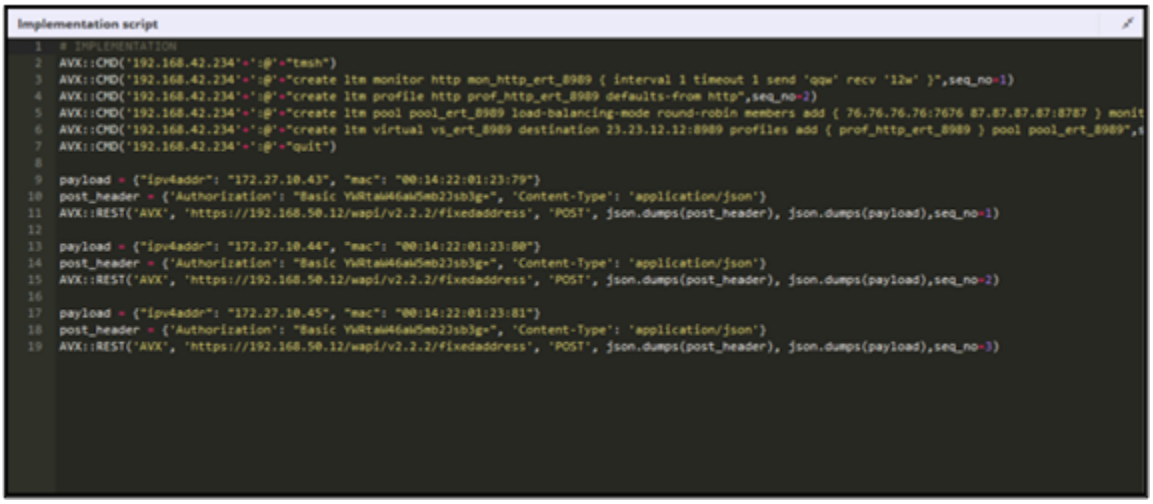
String	Description
command	Actual command to be executed on the device.
interactive	• Used if the command needs to be executed where devices require interactive prompts (such as A10) • Similar to providing a password for SSH.
Seq_no	• Provision to define sequence numbers/tags against configurations (Implementation, 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 © 2025 AppViewX, Inc. All Rights Reserved.

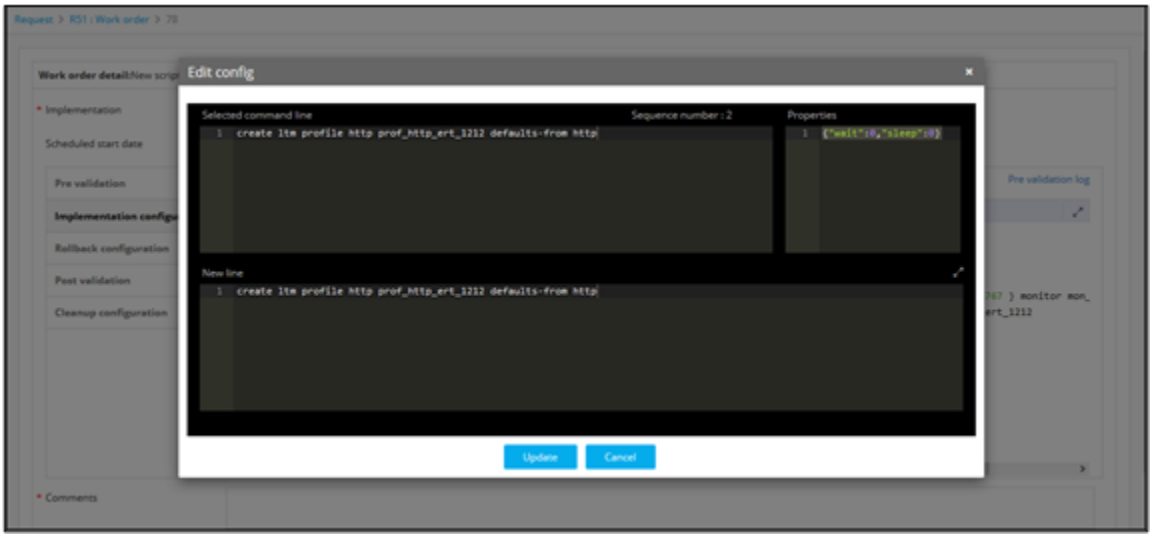
Script with Sequence number/Tags - Implementation Config:



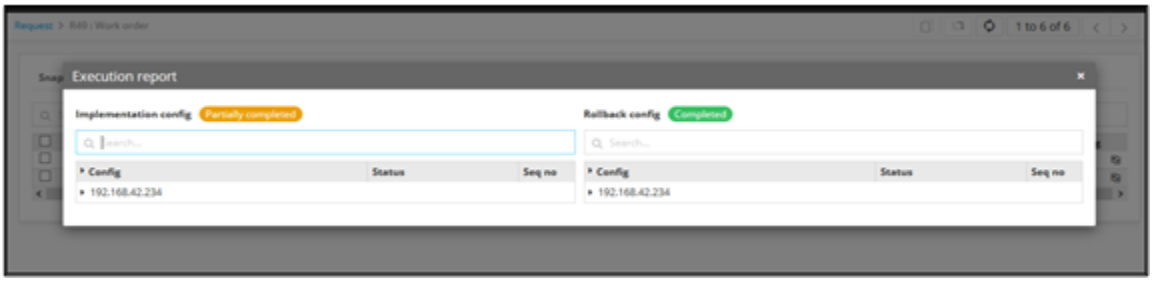
Script with Sequence number/Tags - Rollback Config:

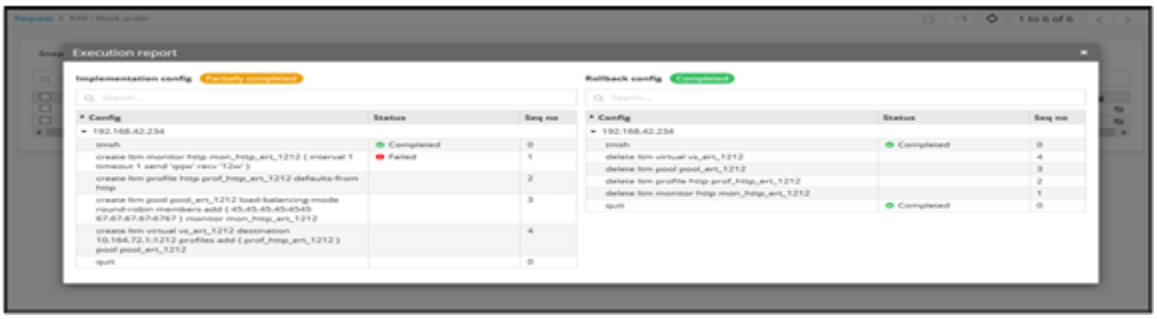
Work order Review pane with option to edit command and properties:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



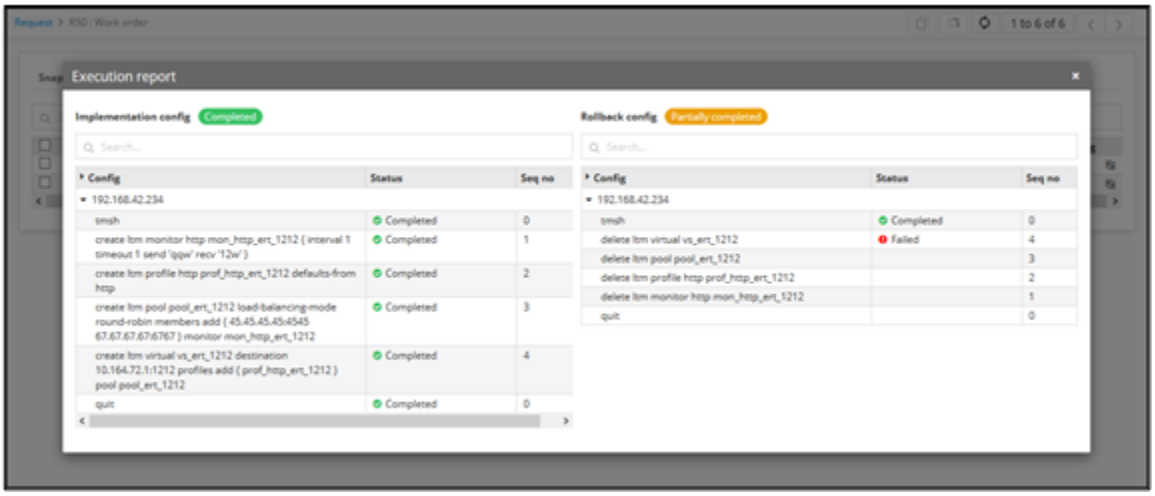
Execution Report - Rollback completion for Partial Implementation:



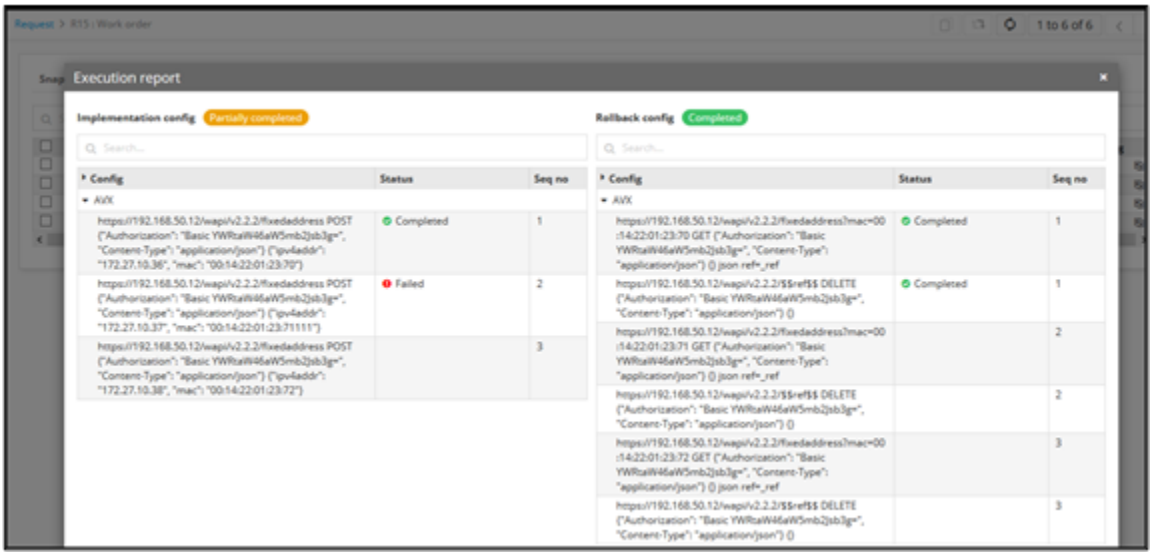


Execution Report for Commands:

Copyright © 2025 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}) Json_data is the json value to be passed to REST

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

URL	Endpoint URL to hit the 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 © 2025 AppViewX, Inc. All Rights Reserved.

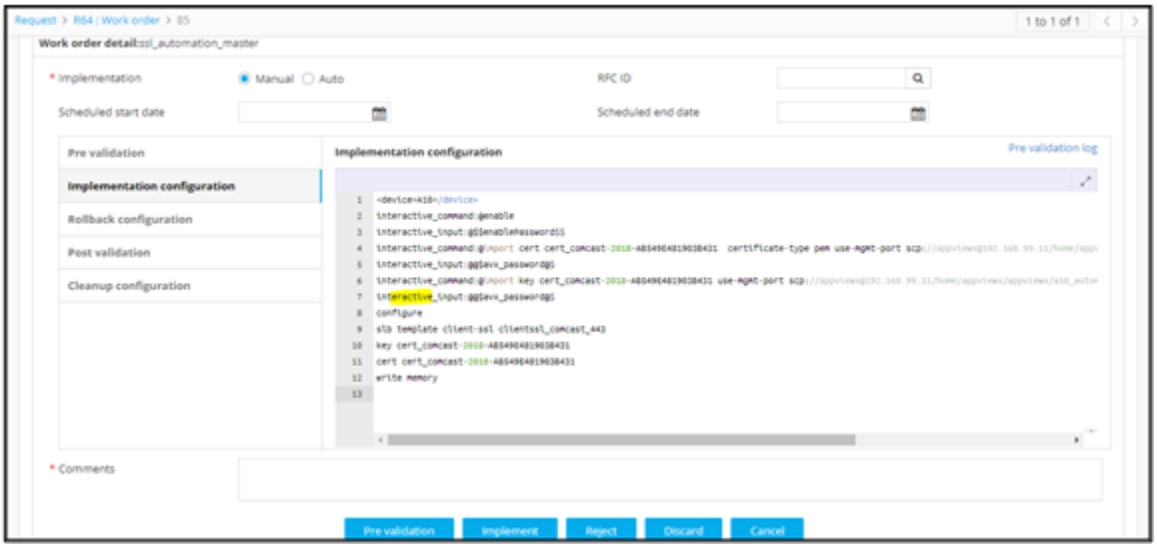
```
AVX::PASSWORD('@$avx_password@$',password)
```

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

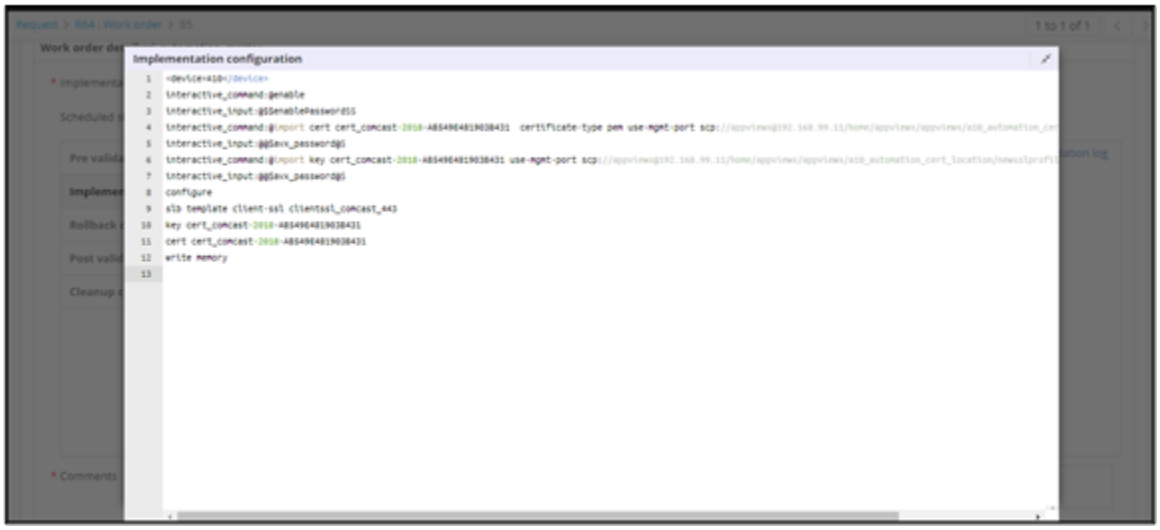
AVX::PASSWORD usage with configuration sample:

```
<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 © 2025 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')
```

String	Description
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.

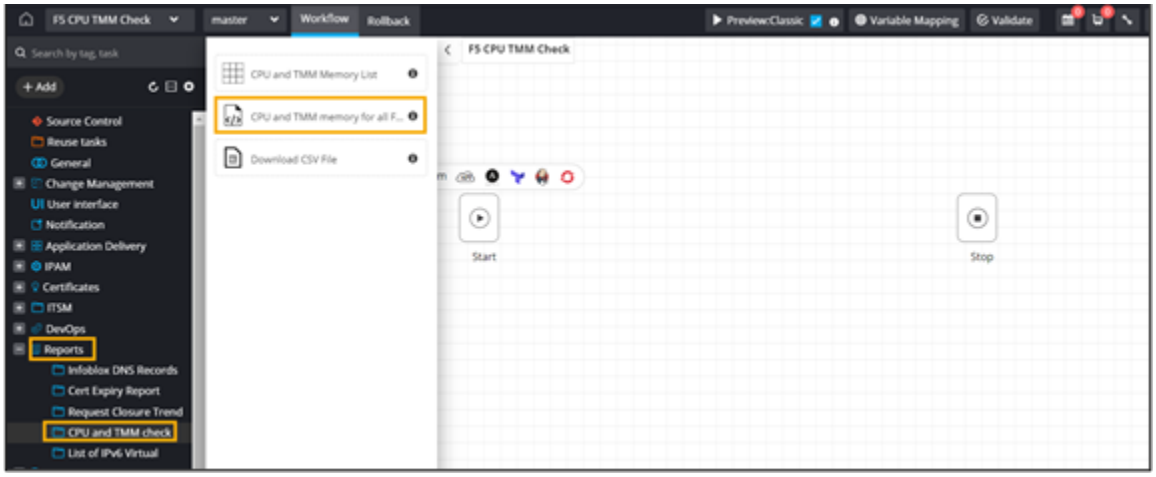
Syntax:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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

To use this command:

1. Design a new workflow.
2. In the deisgn space, from the left menu, under **Reports** > **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.



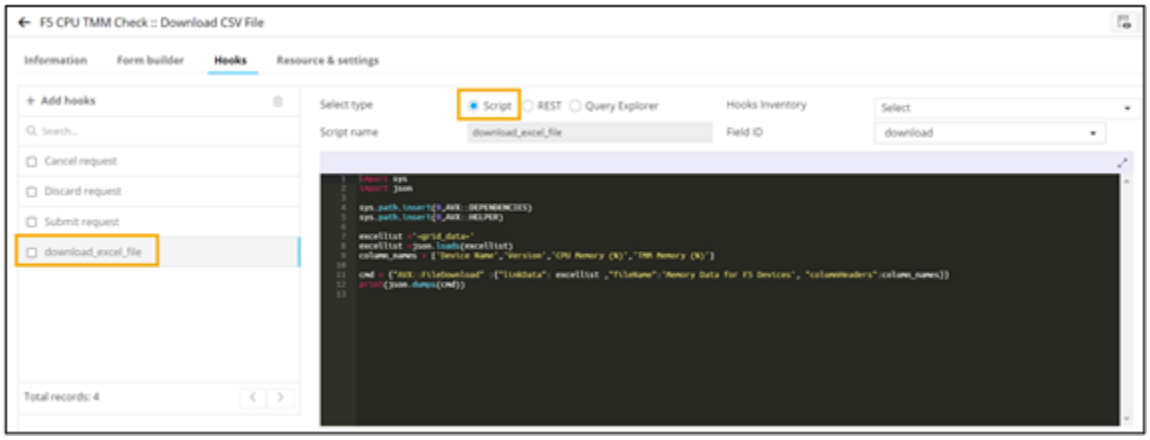
3. From the same folder, drag and drop the **CPU and TMM memory List** task. This task will display the Device name, CPU (%), and TMM (%) in a grid view.

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

4. From the same folder, 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 © 2025 AppViewX, Inc. All Rights Reserved.

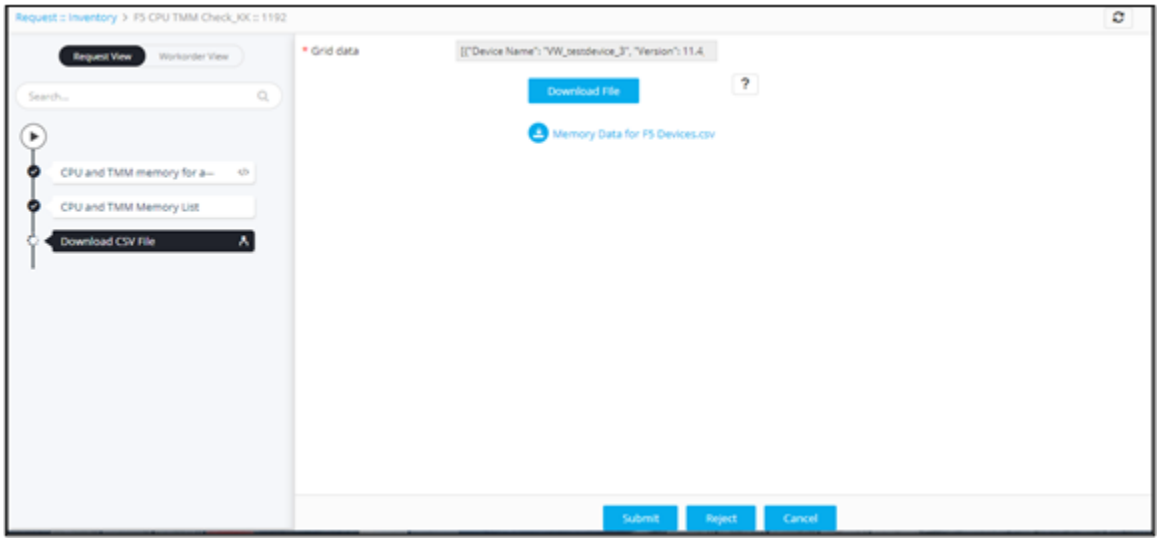
5. In the **Download CSV file** task window, define the file download logic to generate a <.csv> file on run time.




```
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))
```

6. Connect and enable the workflow.
7. Trigger the workflow.
8. At the Download CSV file stage of workflow execution, click **Download File** to generate the .csv file.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The file is downloaded to your machine.

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)
```

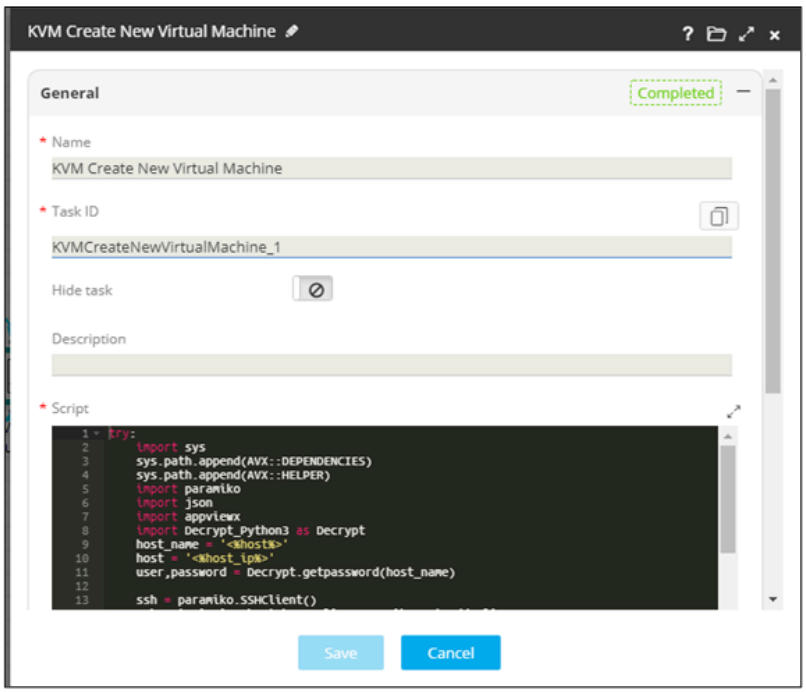
```
mgmt_interface=mgmt_interface,add_interface=add_interface)


  #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:
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

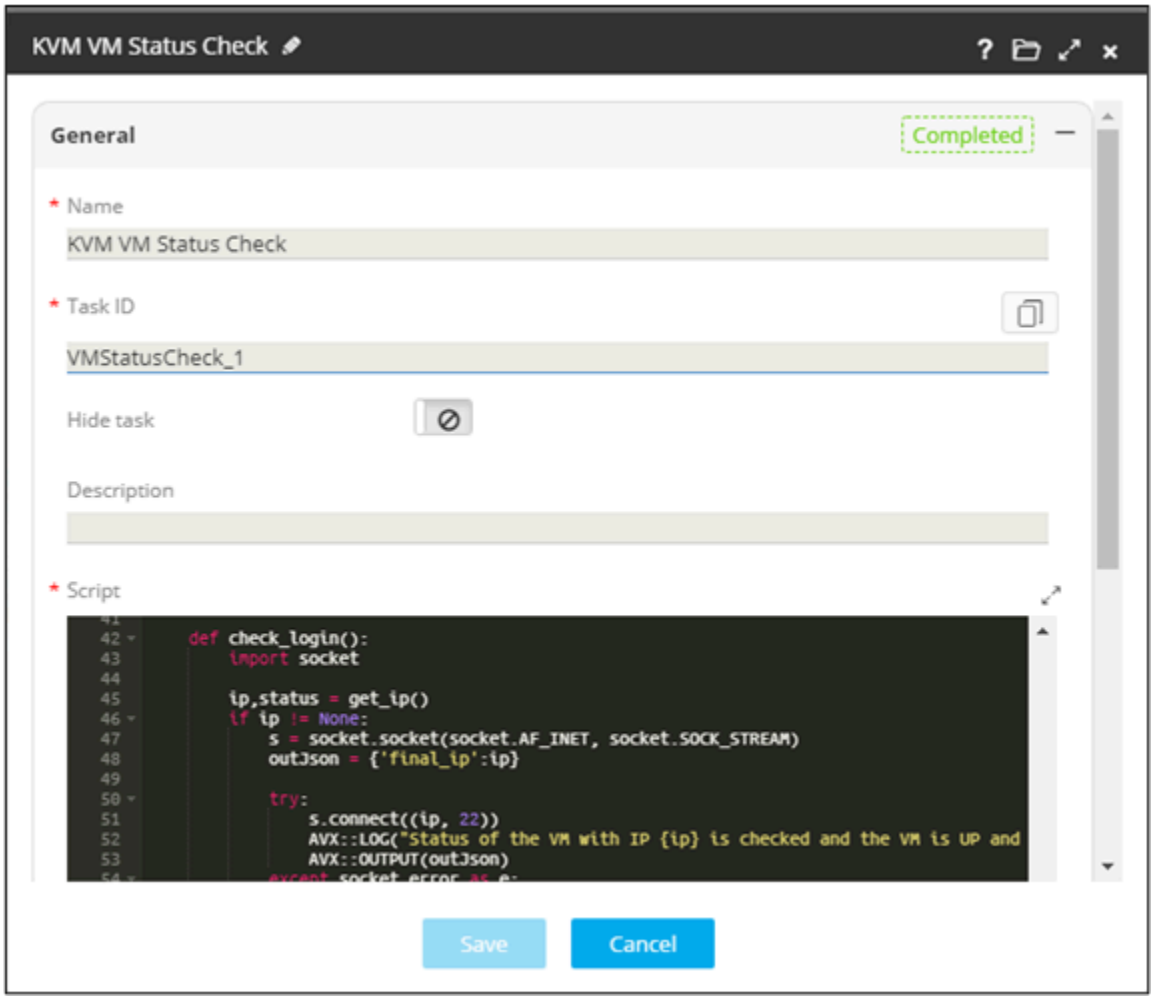
```
        AVX::LOG("Disk and VM Creation Failed"+str(stderr_vm.read().decode()))
        AVX::OUTPUT(outJson,2)
    ssh.close()
```

```
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 © 2025 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
        # dataJson["logs"] = [{"message": "Status of the VM with IP {ip} is checked and the VM is UP and RUNNING".format(ip=ip)}]
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    # 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 pre-validation 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 © 2025 AppViewX, Inc. All Rights Reserved.

OR

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

AVX::ESCAPE

AVX::ESCAPE function has been designed to escape non-alphanumeric characters in user or third-party inputs when processed by an operating system shell. The function loops through each character in the input string and escapes any character that is not alphanumeric. It returns a new string where unsafe characters are prefixed with a backslash (\), ensuring they are interpreted as literal characters rather than part of the shell command.

Usage: Script task when executing a **python script** with special or non-alphanumeric characters.

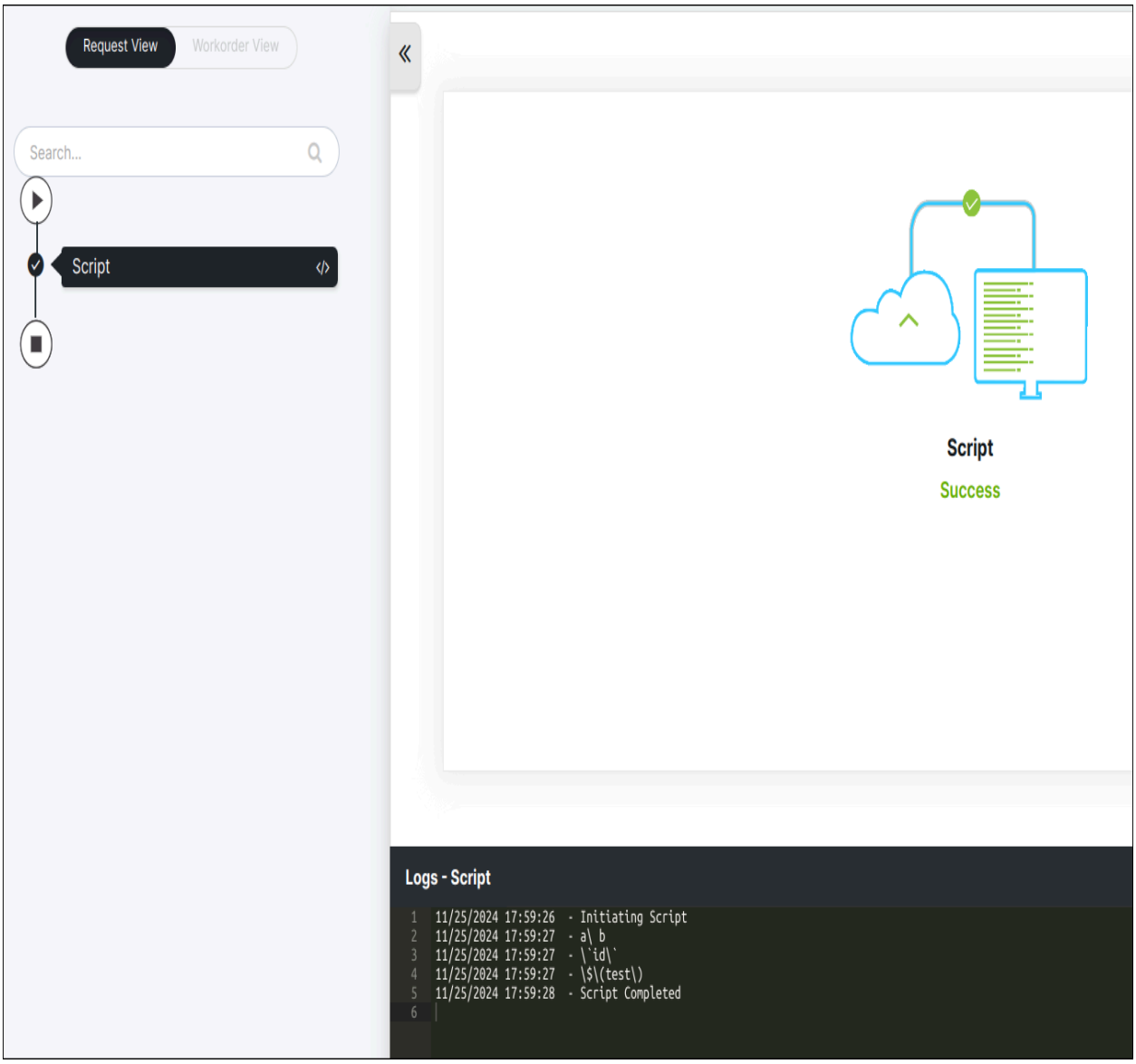
```
variable1 = "a b"

variable2 = "'id'"

variable3 = "$(test)"

esc1 = AVX::ESCAPE(variable1)

esc2 = AVX::ESCAPE(variable2)

esc3 = AVX::ESCAPE(variable3)

AVX::LOG(esc1)

AVX::LOG(esc2)

AVX::LOG(esc3)

AVX::OUTPUT({})
```

The above python script has three variables:

- Variable 1 has a space **"a b"**
- Variable 2 has quotes **"**'id'**"**
- Variable 3 has a $ symbol **$(test)**

The above three cases will be escaped with the backslash (\)when executing the script.

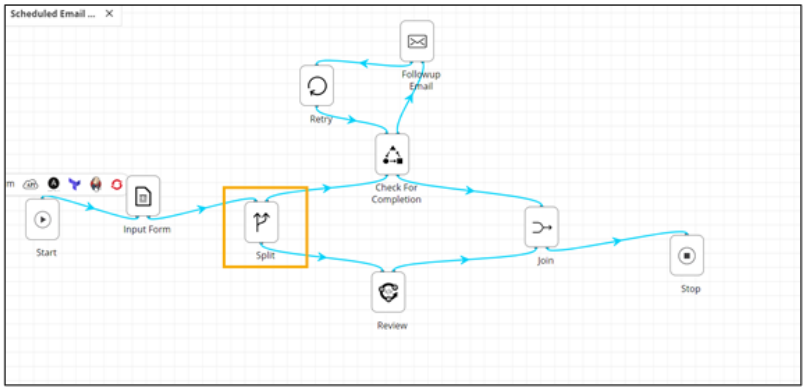
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Split

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

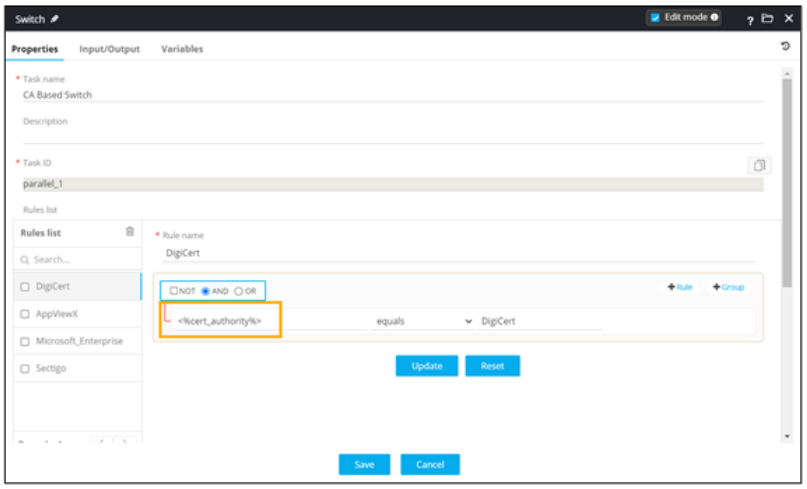


Note: The Join task must be used as a common connector when a Split task is used.

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.



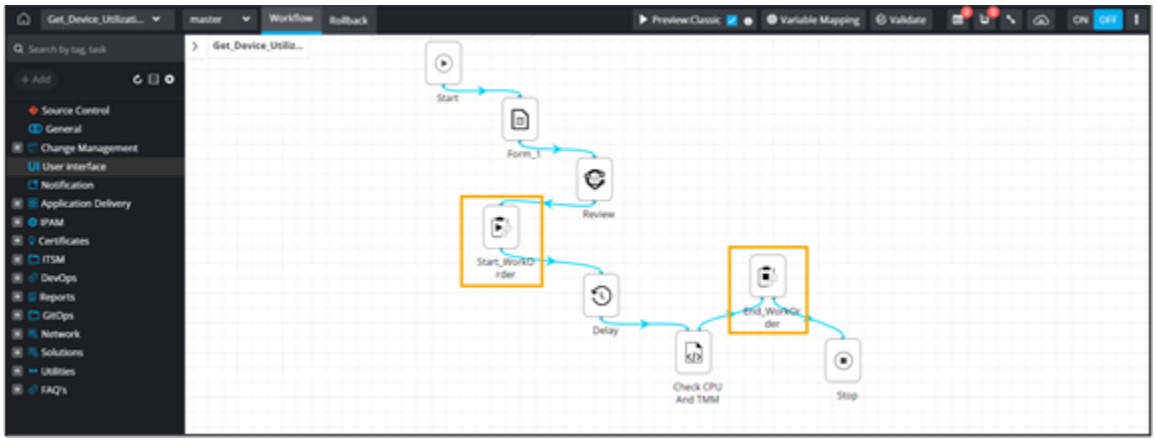
WorkOrder

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.

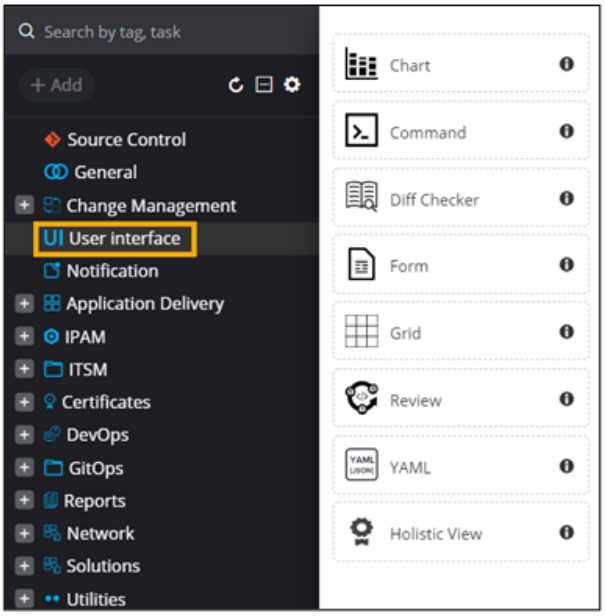
For example, you are executing two implementations, one for a firewall device and the other for a certificate. Now if we just want to rollback the firewall because of some issue in its implementation, then a work order can be used in this situation.



Task Category - User Interface

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Access levels for Resources

The user interface (UI) palettes can be configured for user groups or individual users to have the following access levels

- **V - View:** Users with View permission can view the task and will not be allowed to submit it.

> **Note:** “View” is not applicable for the form task.

- **R - Review:** Users with Review permission can view a particular user interface task and submit it, but they cannot make any changes to the request generated.
- **RW - Read Write:** Users with Read Write permission can modify the particular user interface task and submit it.

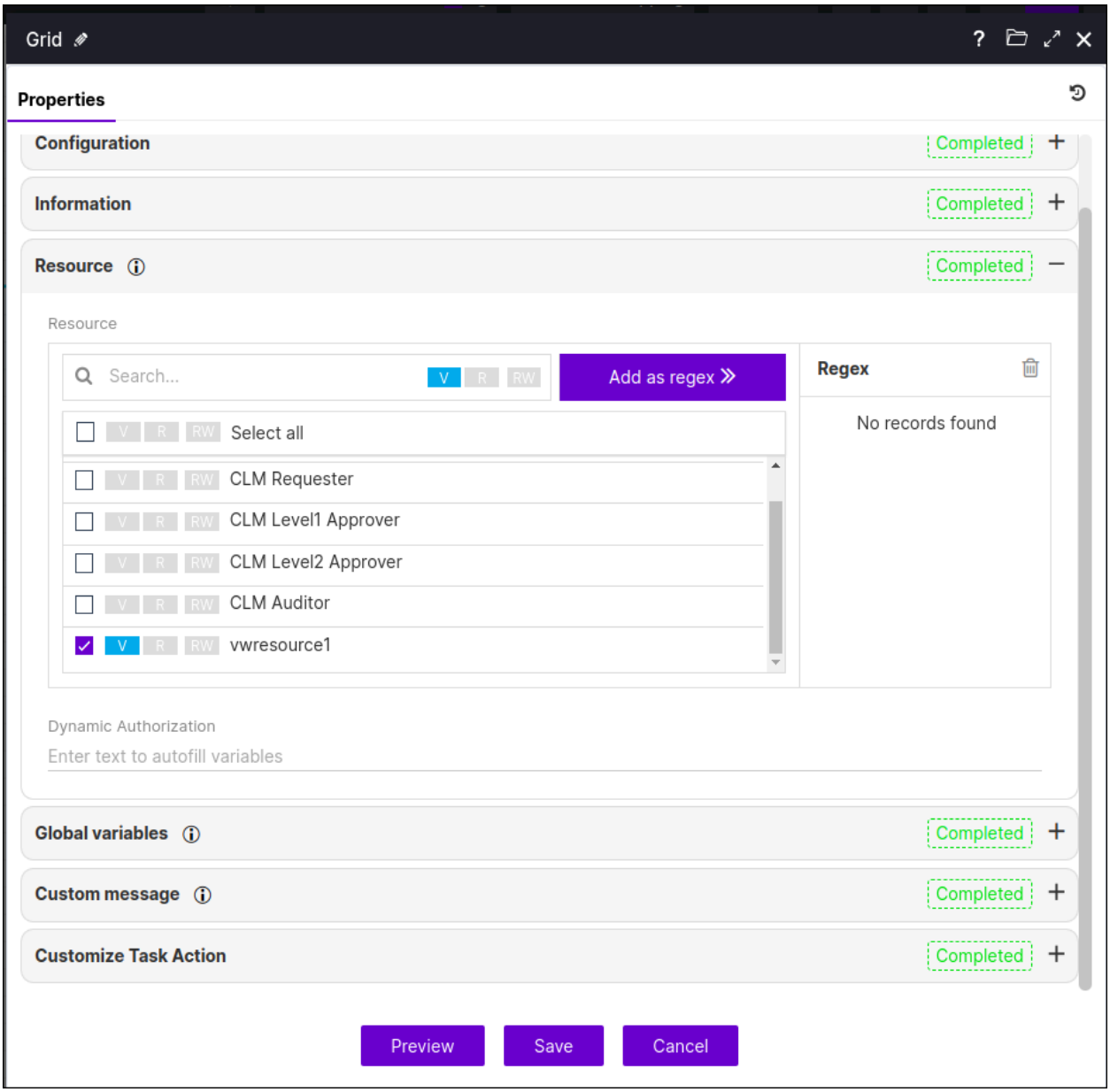
To configure the resource access,

1. Design a workflow.
2. From the **General** section, drag and drop the **Form** task.
3. Add the required form fields as shown 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.
6. In the **Grid** task window, under **Resource**, define the permissions **V**, **R**, **RW**.
7. Connect the workflow tasks and from the menu on the top, click **Preview**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

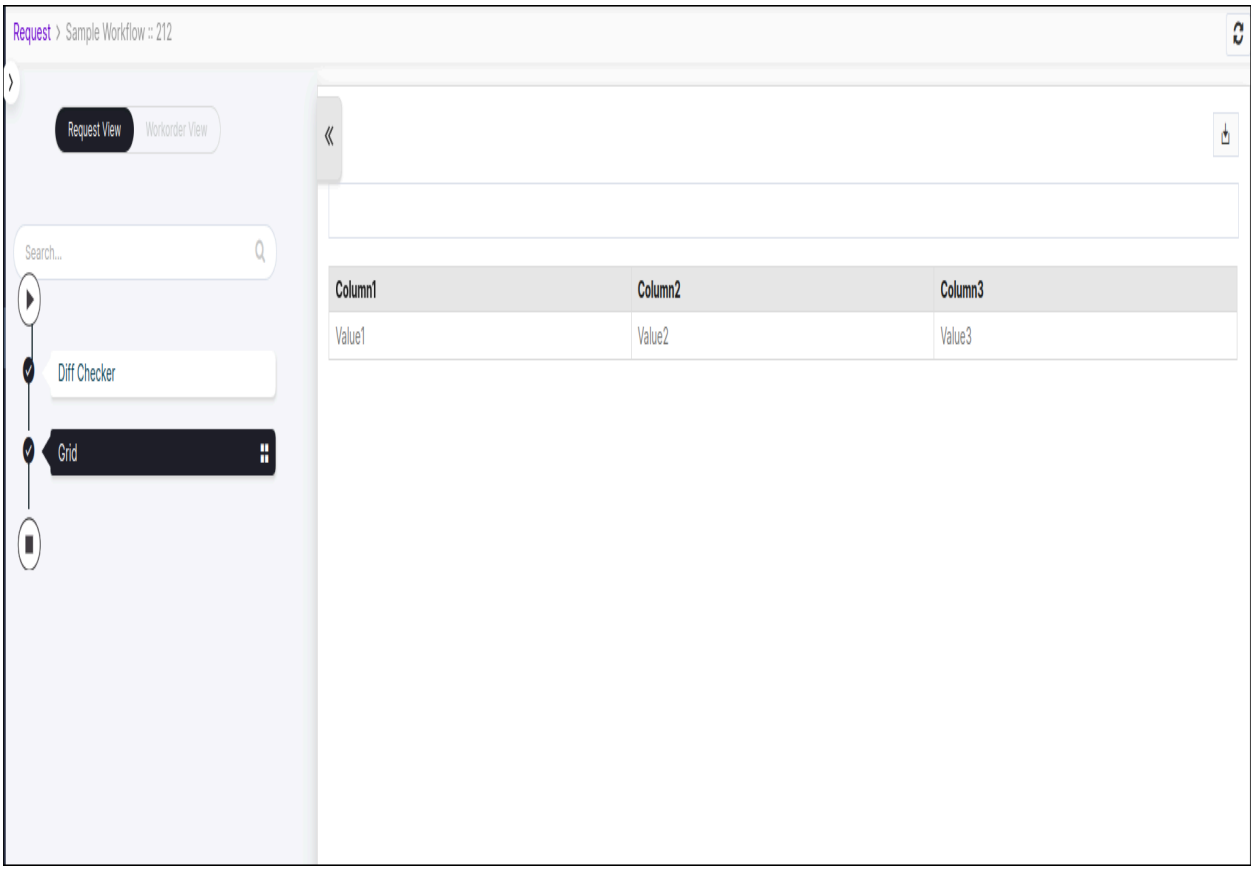
Use case: When a resource is assigned a view permission , they will not be allowed to submit or cancel the request.

In the below grid task , **vwresource1** is given only view permission.



When executing the request with above grid task , users will not be able to submit that grid task, but only view the request.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



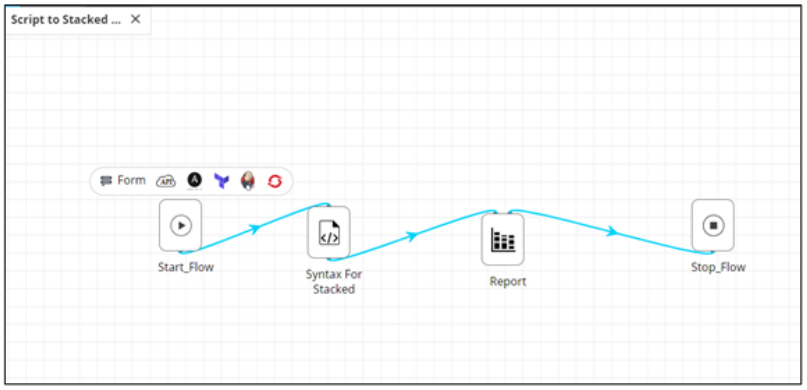
- Chart
- Command
- Diff Checker
- Form
- Grid
- Review
- YAML
- Holistic View

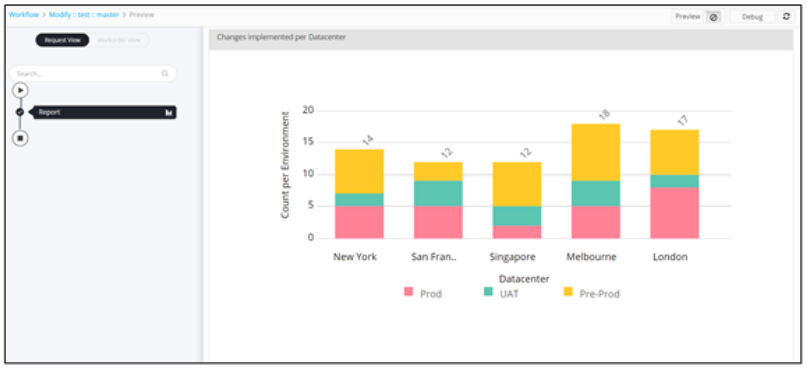
Chart

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.





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

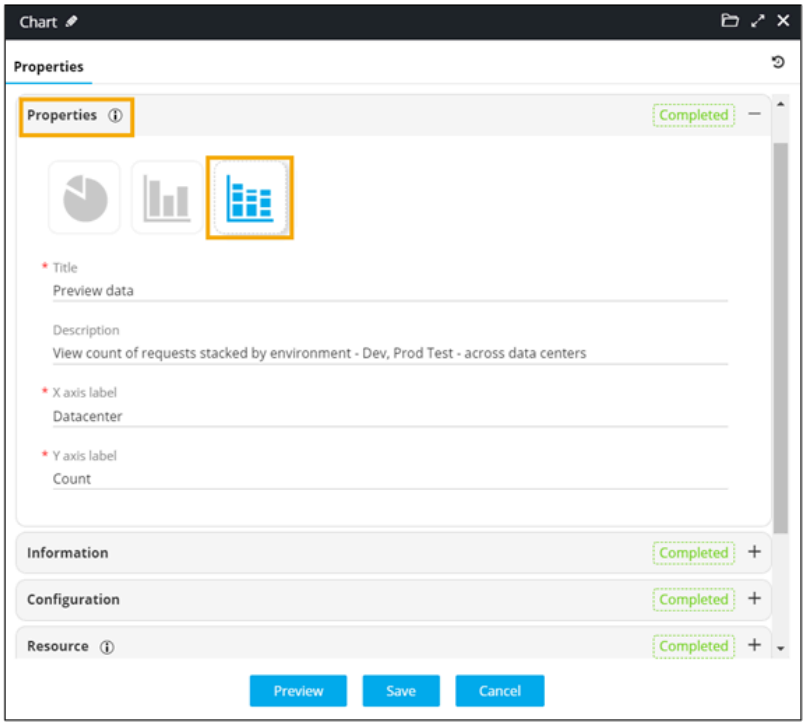
Designing a Chart with Static Data

To design a stacked chart using static data:

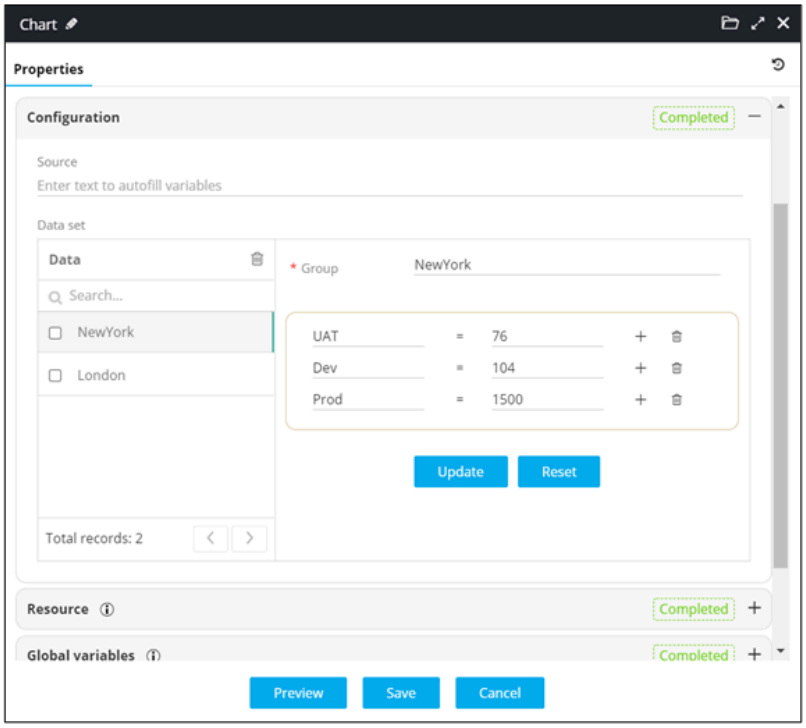
1. Design a workflow.
2. From the User Interface section, drag and drop the **Chart** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.



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



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

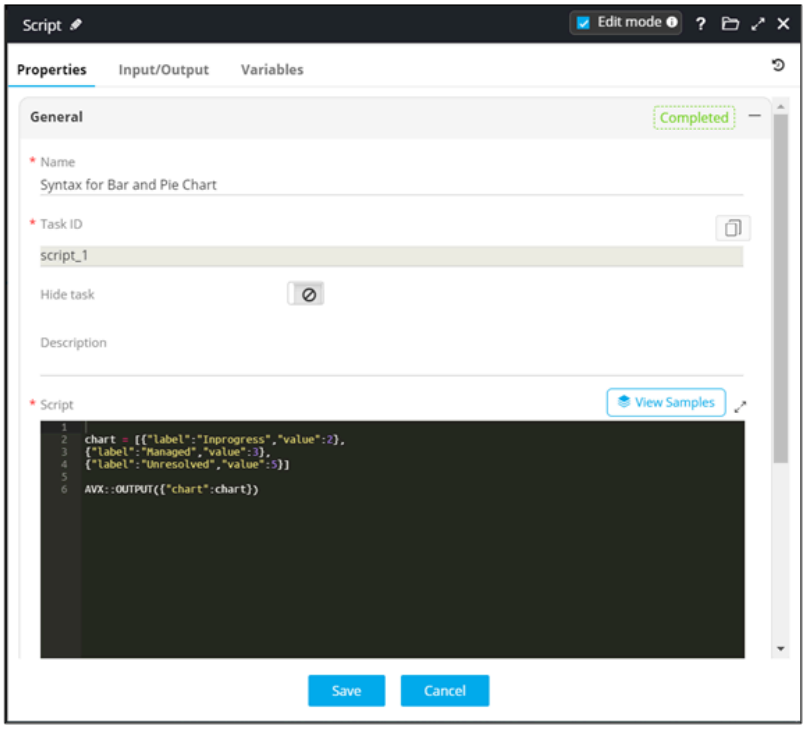
6. In the **Chart** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
7. Click **Save**.
8. Connect the workflow tasks.
9. From the command bar on the top, click **Preview**.

   Stacked chart with static data is displayed.

Designing a Chart with Dynamic Values

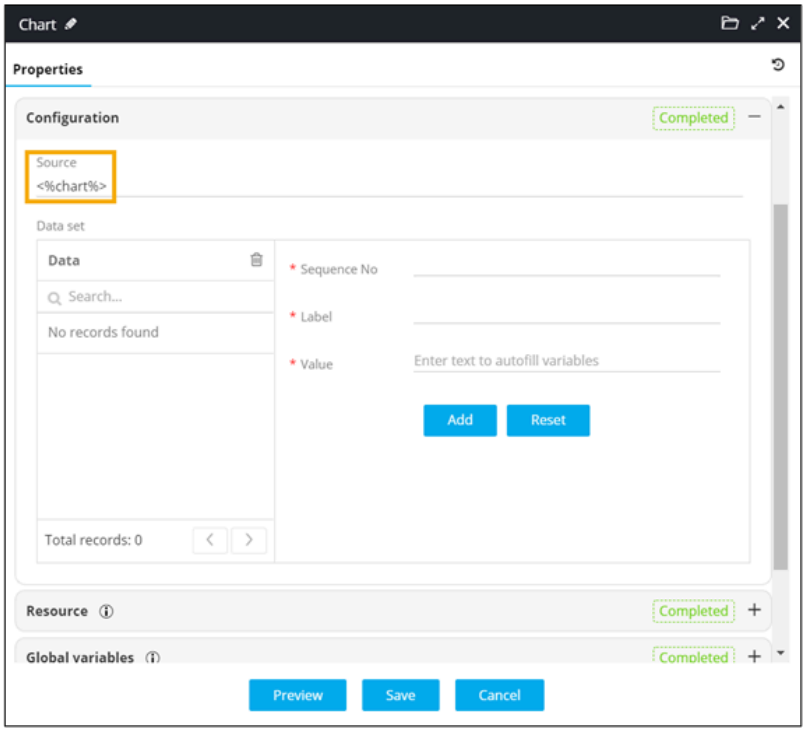
1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Properties**, define a **Script** to get a list of values for the chart.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

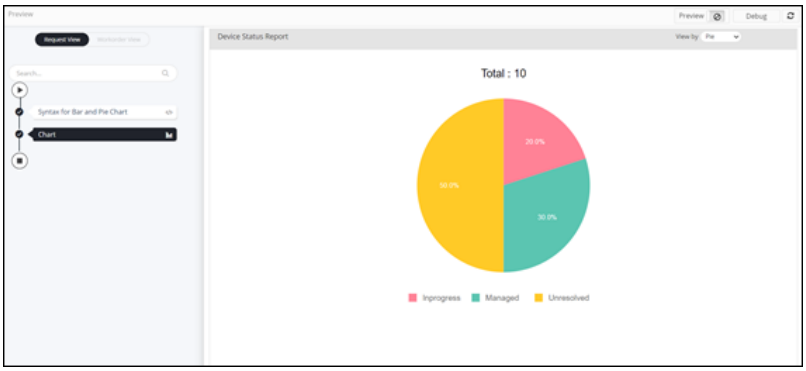


4. From the **User Interface** section, drag and drop a **Chart** task.
5. In the **Chart** task window, under **Properties**, select the chart type as Pie and enter or select the field information.
6. Enter the chart **Title** (mandatory field) and **Description** and select the **Legend position**.
7. Under **Configuration**, refer the global variable to render data.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



8. Connect the workflow tasks and from the menu on the top, click **Preview**.
   Chart with dynamic values is displayed.



Note: You can view this chart as a **Donut** chart by selecting the option from the **View by** dropdown.

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 configuration automation using SSH and CLI.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

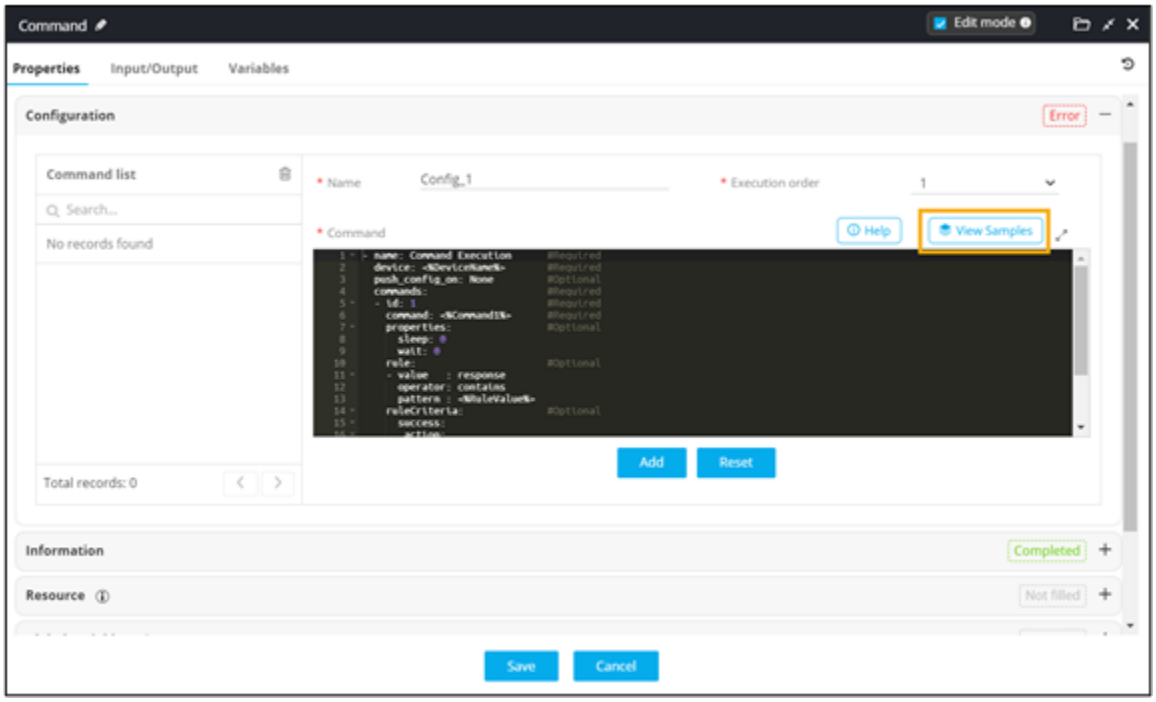
- YAML - declarative command execution
- Parallel, sequential configuration execution
- Intelligent command execution with in-built rules (Multi-boolean conditions operators with shortcuts)
- Event handling and dynamic variable support
- Modify configurations and reviews
- Pre-validation and post-validation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

To use a Command task within your workflow:

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

Using Command Task in the Workflow

1. Design a workflow.
2. From the **User Interface** section in the left menu, drag and drop the **Command** task.
3. To use a prebuilt command sample, in the **Command** task window, under **Properties**, in the **Configuration** section, click **View Samples**.

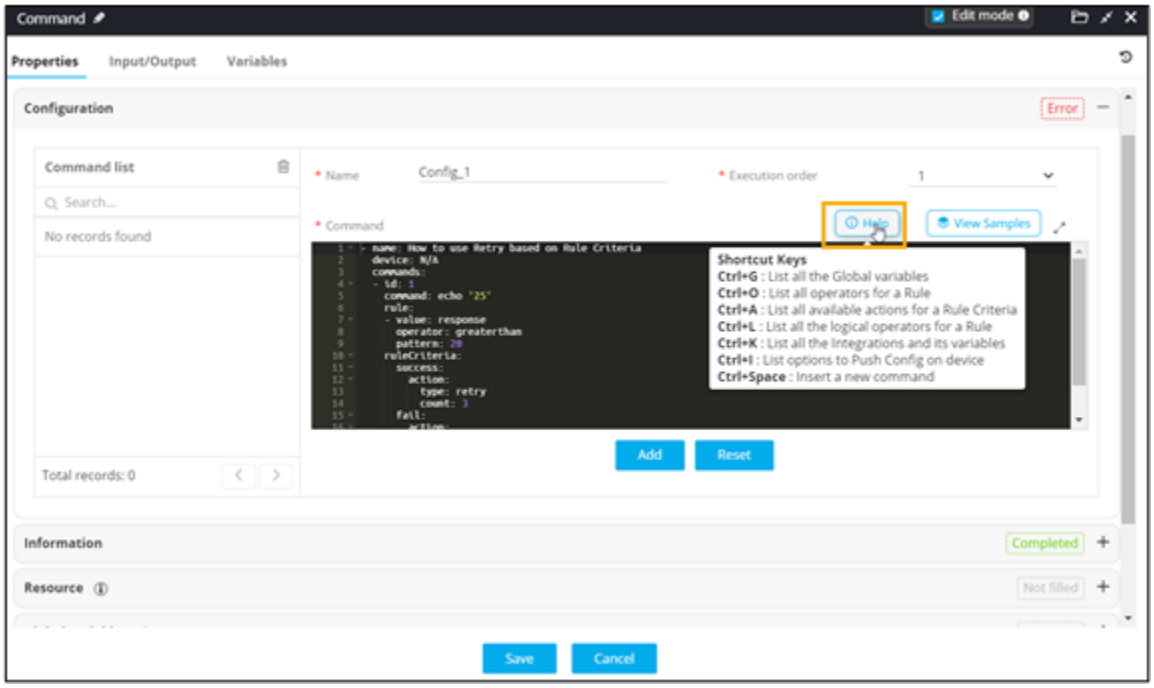


4. In the **Command Samples** window, click **View** next to the Command Sample.
5. To use this configuration, click **Use Sample**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

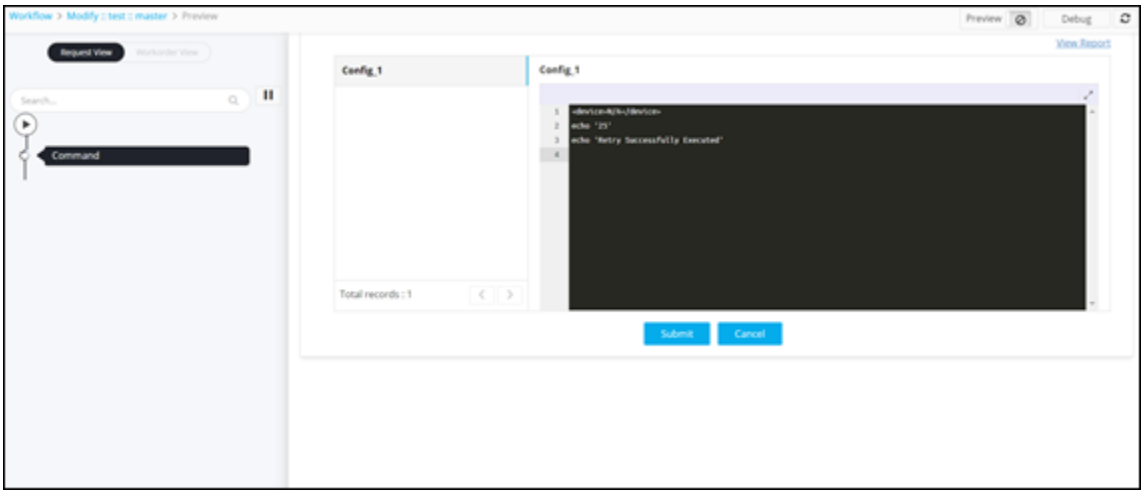
6. To save the configuration settings, in the **Command** task window, under **Properties**, in the **Configuration** section, click **Add**.

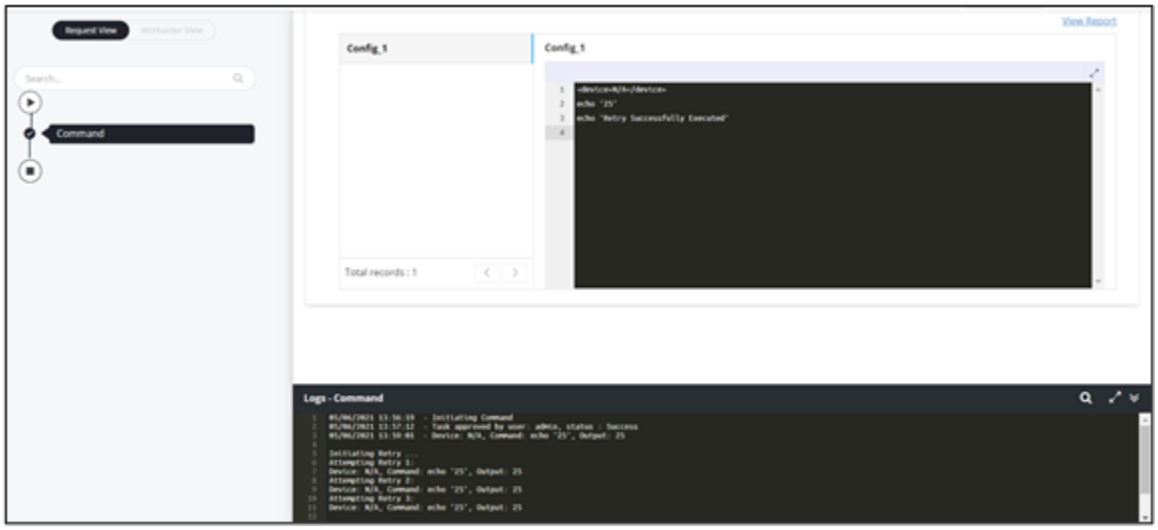
> **Tip:** To see a list of keyboard shortcuts, hover your mouse over **Help**.



7. In the **Command** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
8. Click **Save**.
9. Connect the workflow tasks and from the menu on the top, click **Preview**.
   Command task is executed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.





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

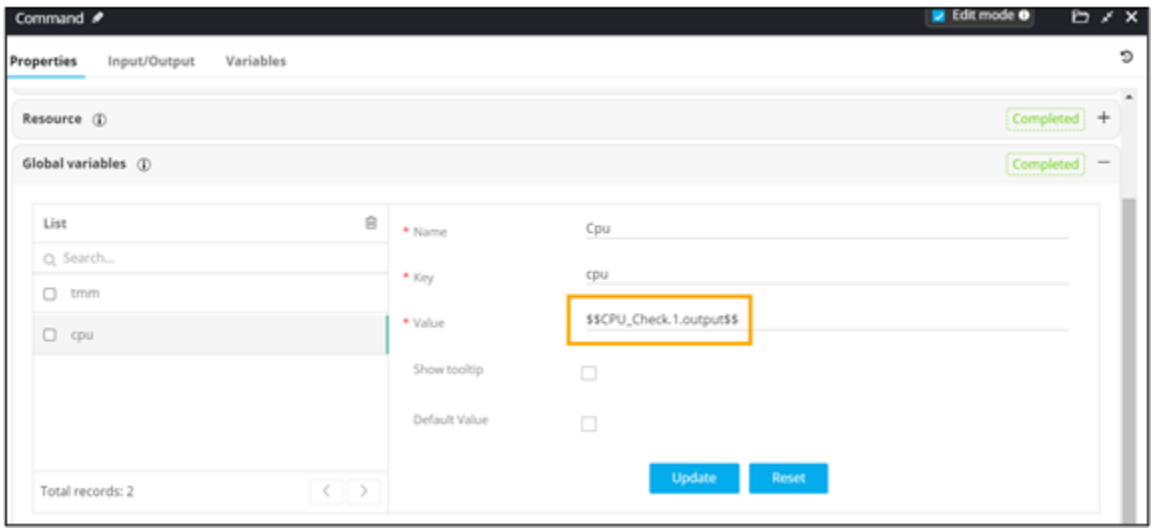
The Command task can be used to pass the response of a task as a global variable for the next task in the workflow. The global variable can be declared in the following ways:

- `$$<Config Name>$$` - 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.
- `$$<Config Name>.<Command ID>.command$$` - 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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: `$$Cpu.1.output$$` `$$Cpu.2.output$$`

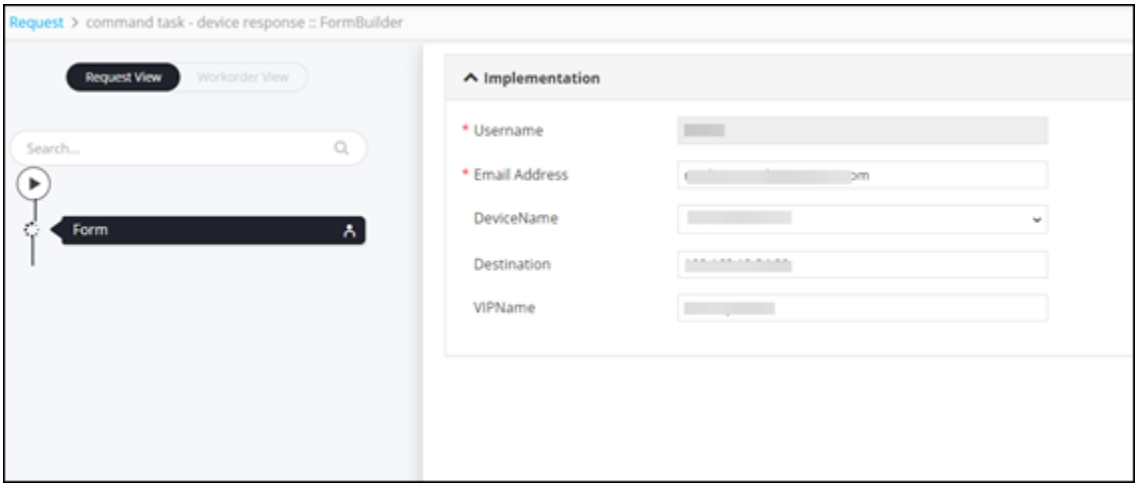


3. In the **If** task window, define the rules for executing the next task in the workflow using the Global variables defined in the Command task.
   For example, create a device if the CPU/TMM values are less than 25.

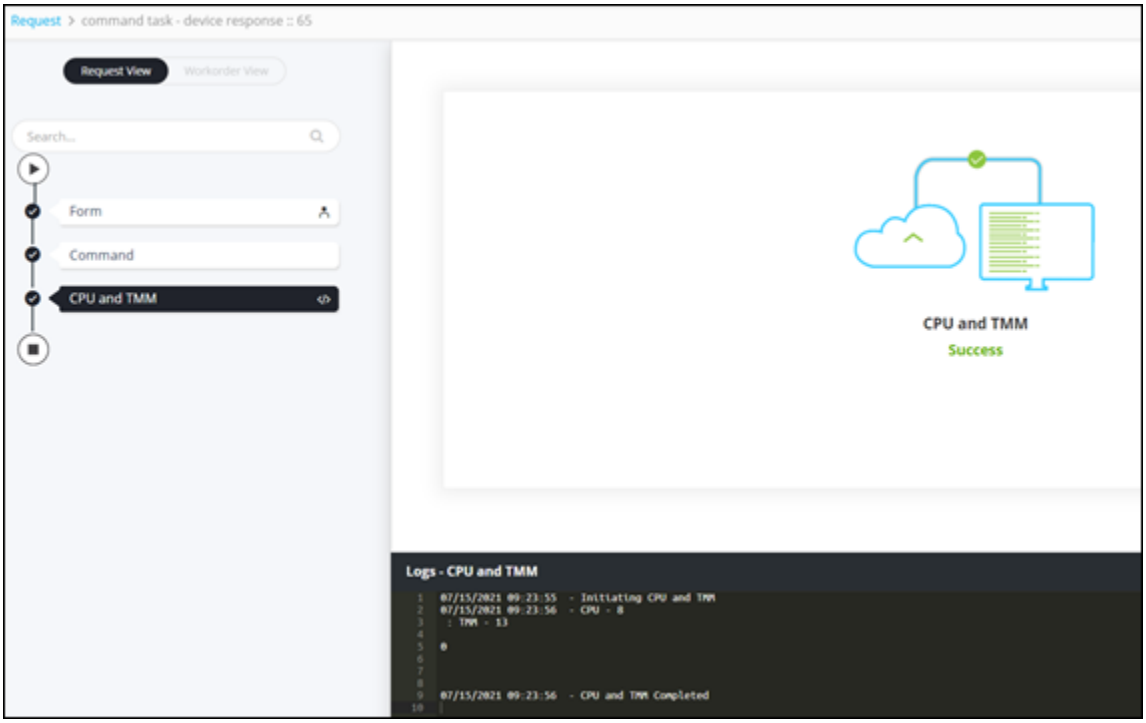
4. Connect the workflow tasks and enable the workflow.
5. Trigger the workflow.
   The workflow execution page is displayed with the workflow inputs requested at the first stage.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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



Command task initiated and CPU and TMM check completed.

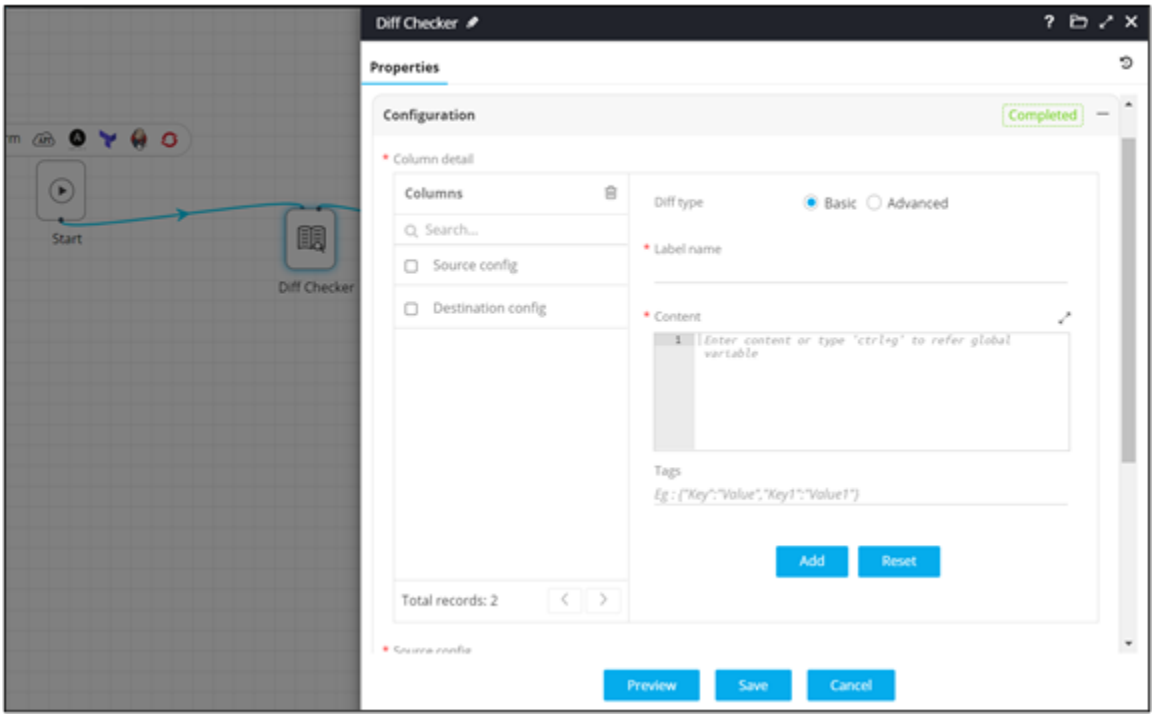


Diff Checker

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 © 2025 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.

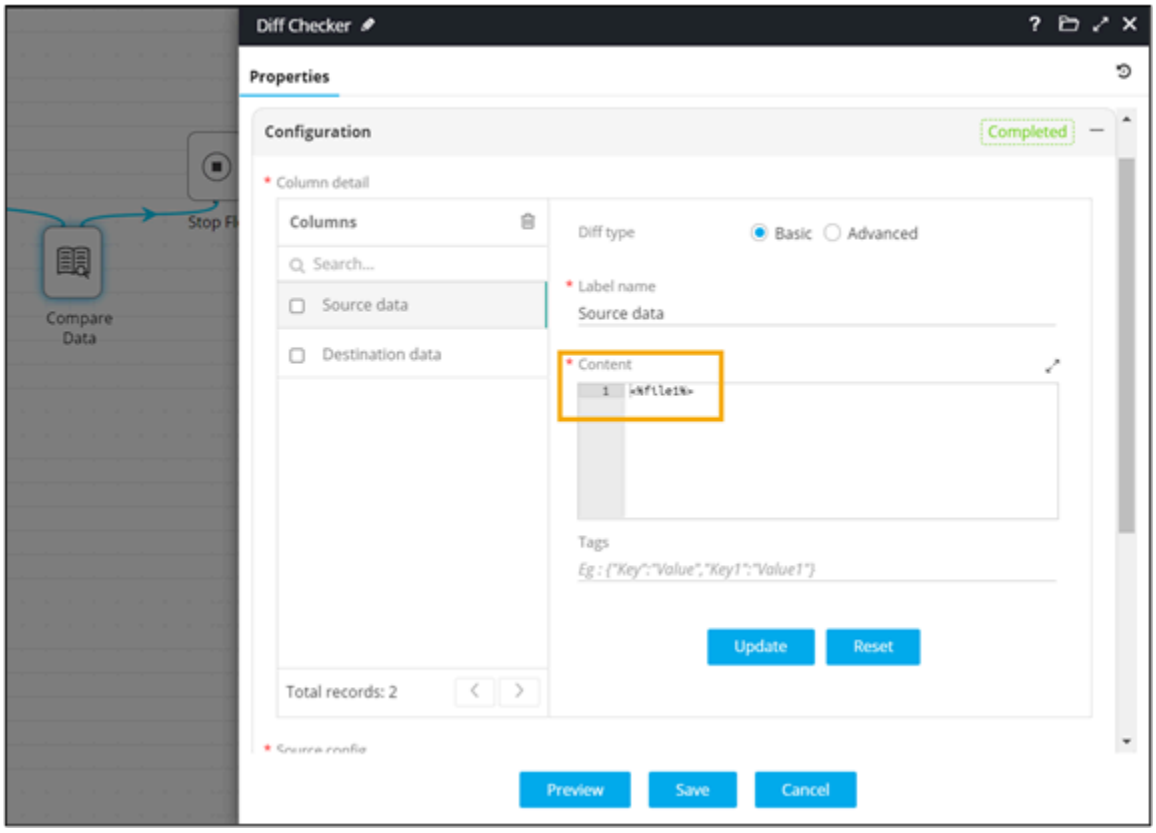


- Configuring a Basic Diff Checker
- Configuring an Advanced Diff Checker

Configuring a Basic Diff Checker

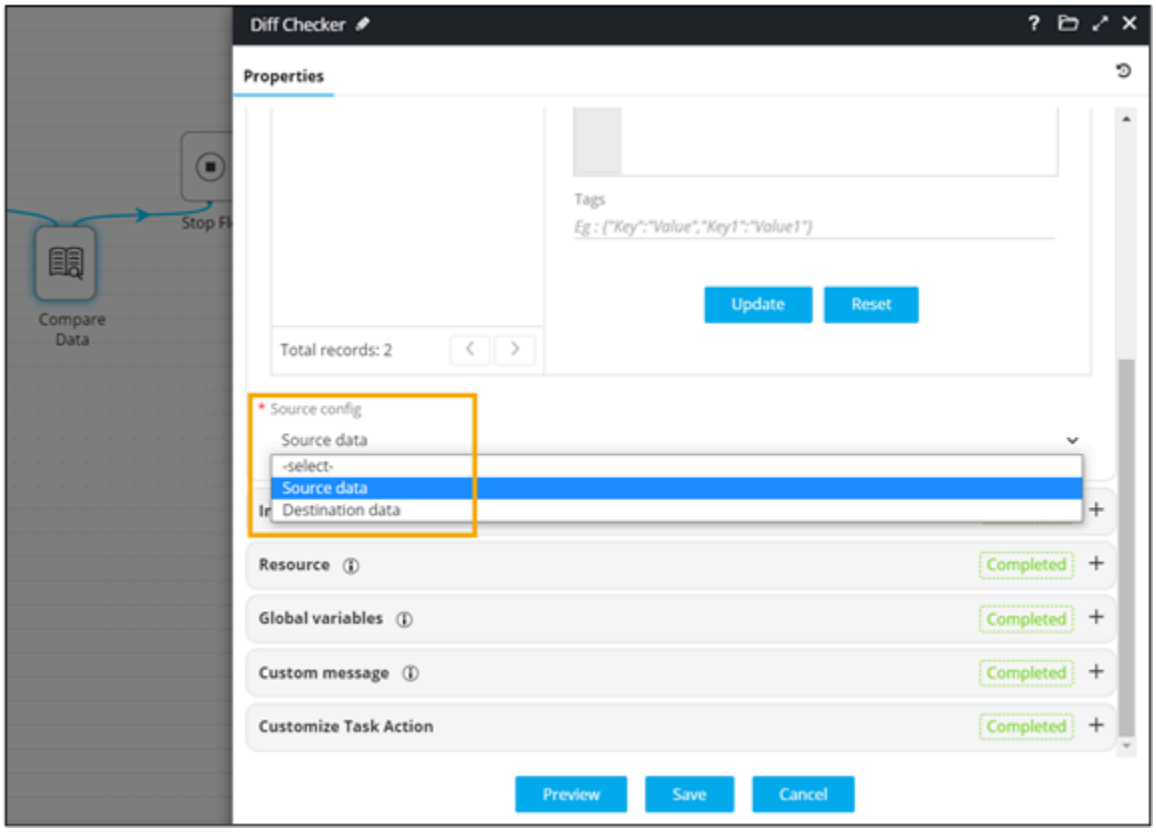
1. Design a workflow.
2. In the design workspace, from the User Interface section, drag and drop the **Diff Checker** task.
3. In the **Diff Checker** task window, under **Properties**, in the **Configuration**section, select **Diff type** as **Basic**.
4. Enter the **Content** to be compared or reference content from any previous workflow task as a variable.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

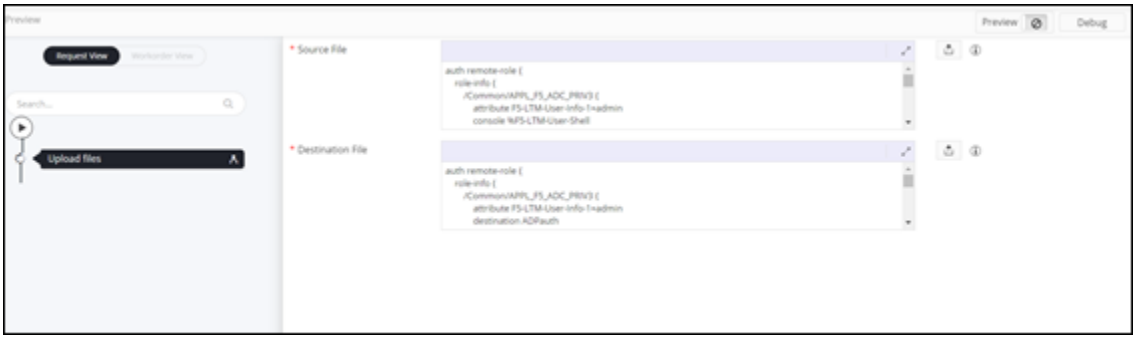


5. Under **Source config**, select the source column against which the diff checker comparison must be made.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. In the **Diff Checker** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
7. Connect the workflow tasks.
8. From the command bar on the top, click **Preview**.
9. Upload the source files.

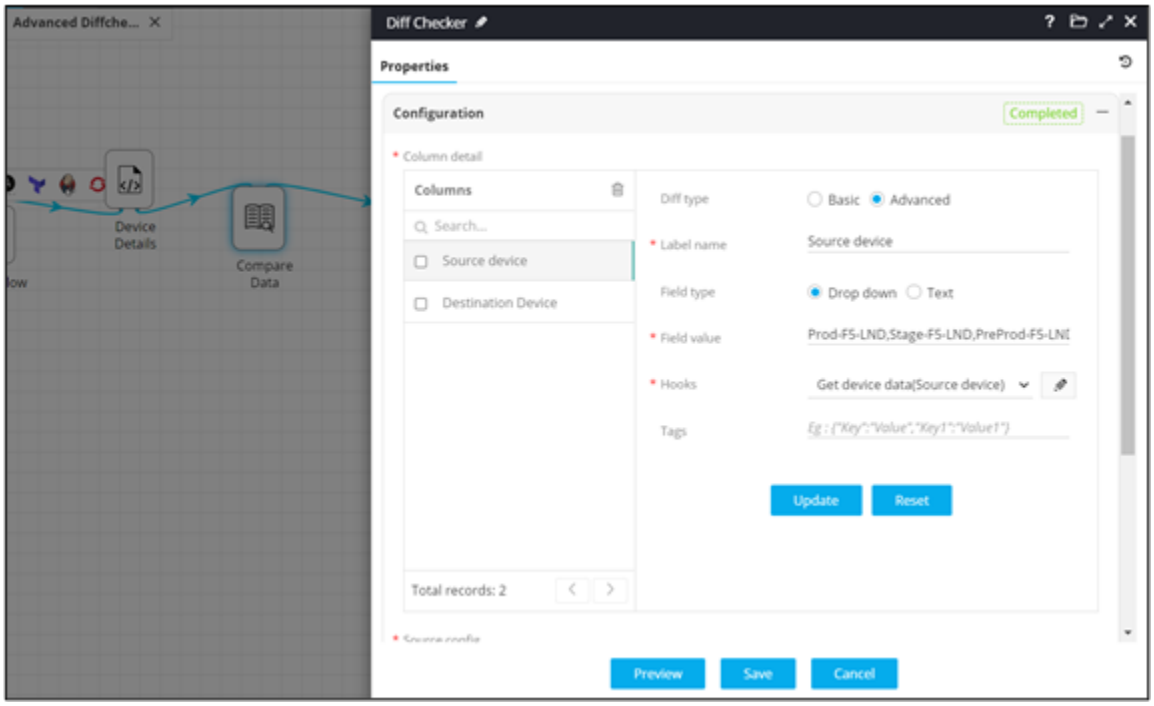


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Basic Diff Checker allows you to compare the data.

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.

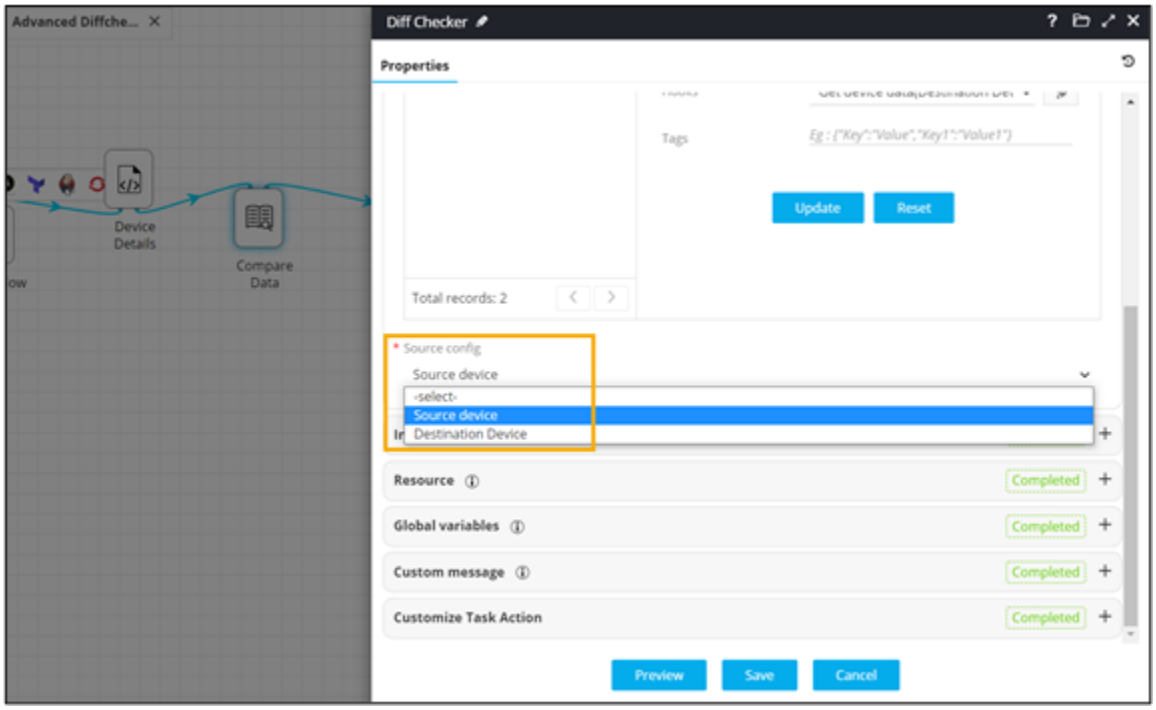


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Advanced section

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.
** : Mandatory fields*	

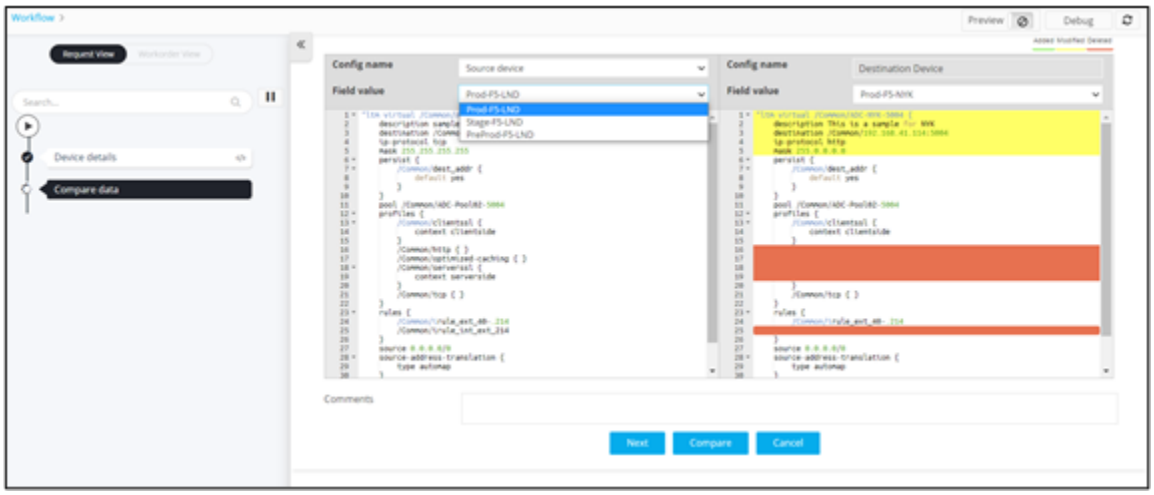
5. Under **Source config**, select the source column against which the diff checker comparison must be made.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. Connect the workflow tasks.
7. From the command bar on the top, click **Preview**.

   Advanced Diff Checker allows you to compare the data.



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.

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

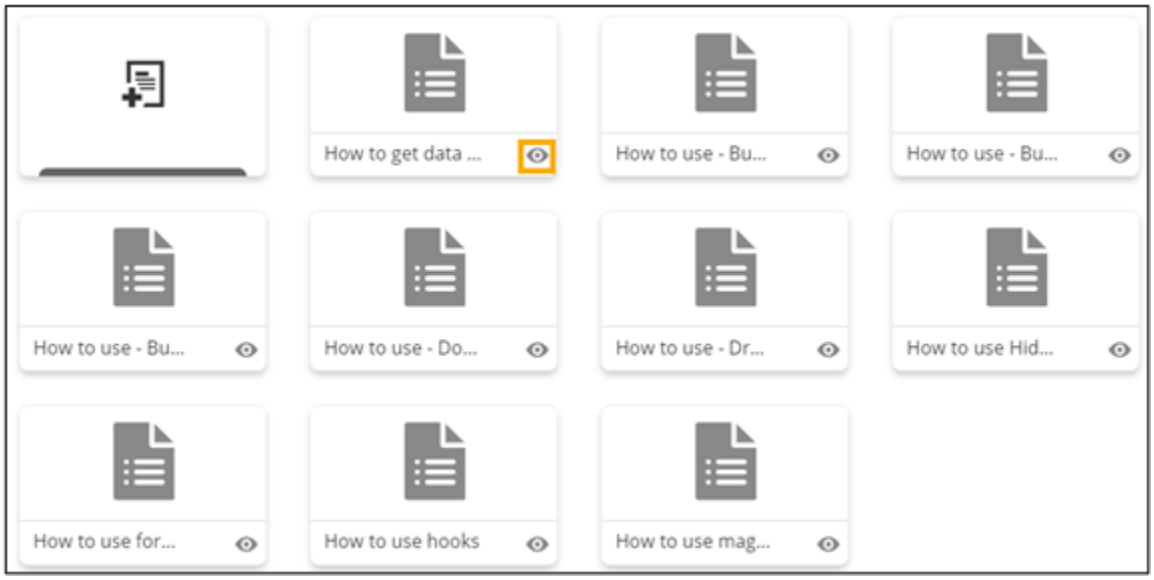
Out of the box Forms

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

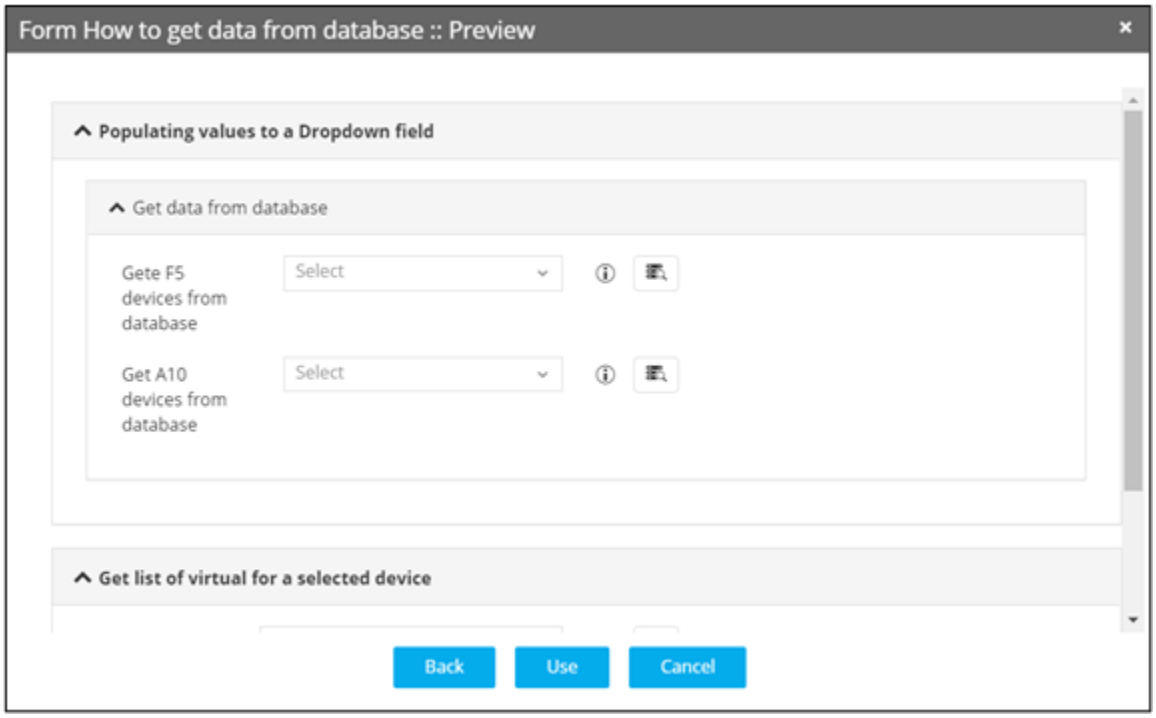
To use an OOB form:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Form** task.
3. In the **Gallery** window, to preview the form, click **(View)** icon on any of the forms.



A preview of the form is displayed.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

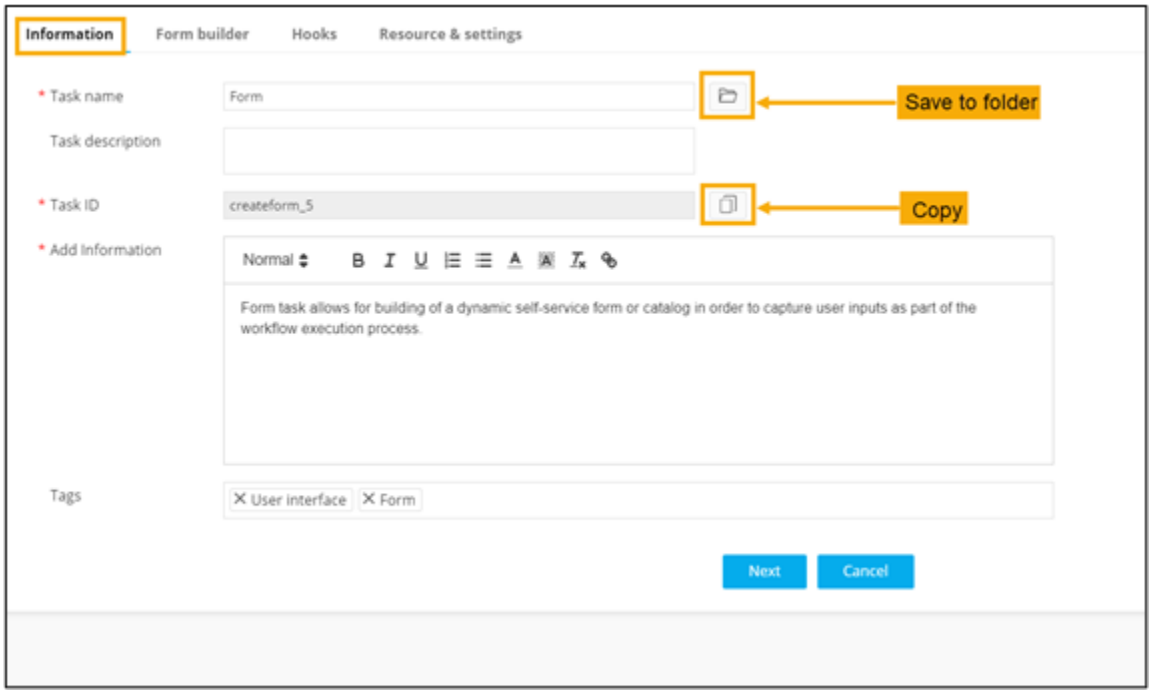
Designing a Custom Form

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.
3. To design a new form, in the **Gallery** window, click **(Create)** icon.
4. Under **Information**, enter the necessary field values.



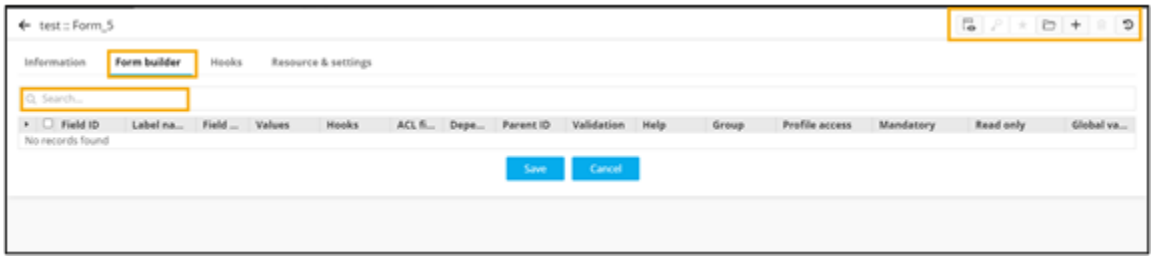
Field descriptions for Information

Field	Description
***Task name**	Provide a unique task name.
	Provision to bookmark the task within a custom folder in the workflow studio.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
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.
** : Mandatory fields*	

5. Under **Form builder**, add the form fields.



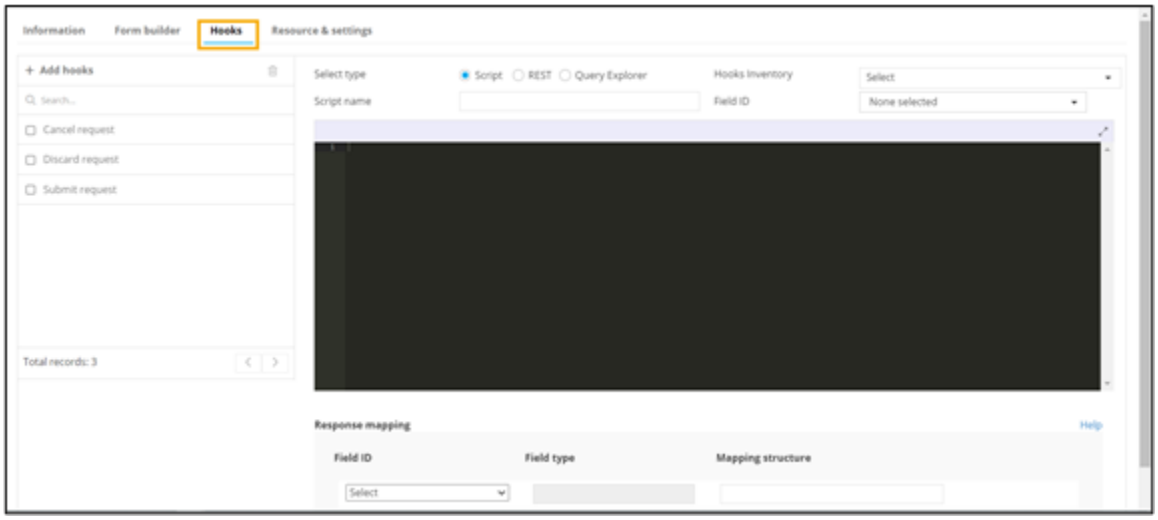
Field descriptions for Form builder

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.

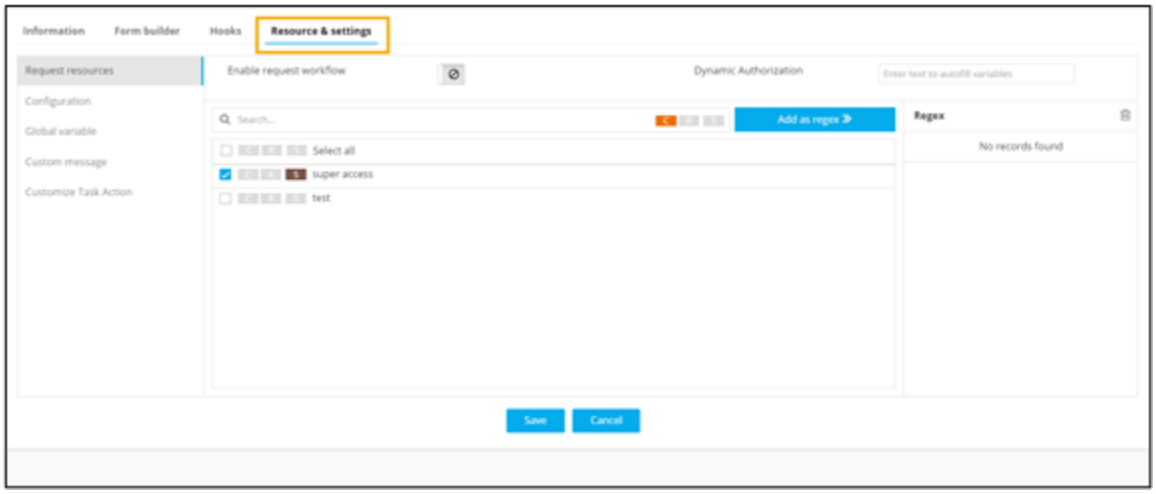
Note: For more information on adding form fields, click here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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**.



7. To assign access to specific roles(s), click **Resource & Settings**.



When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.

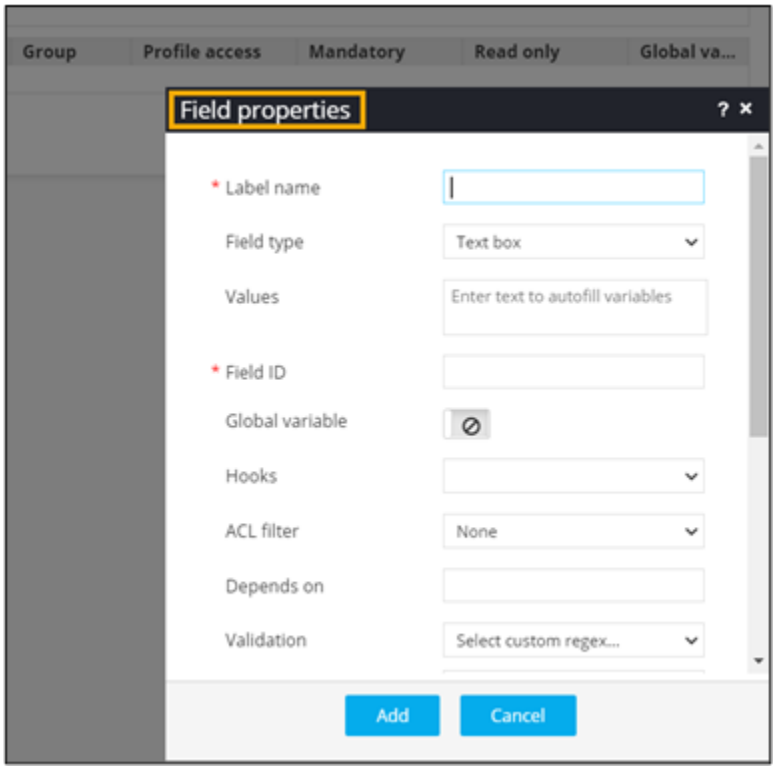
Adding Form Fields in the Form Builder

To add a form field:

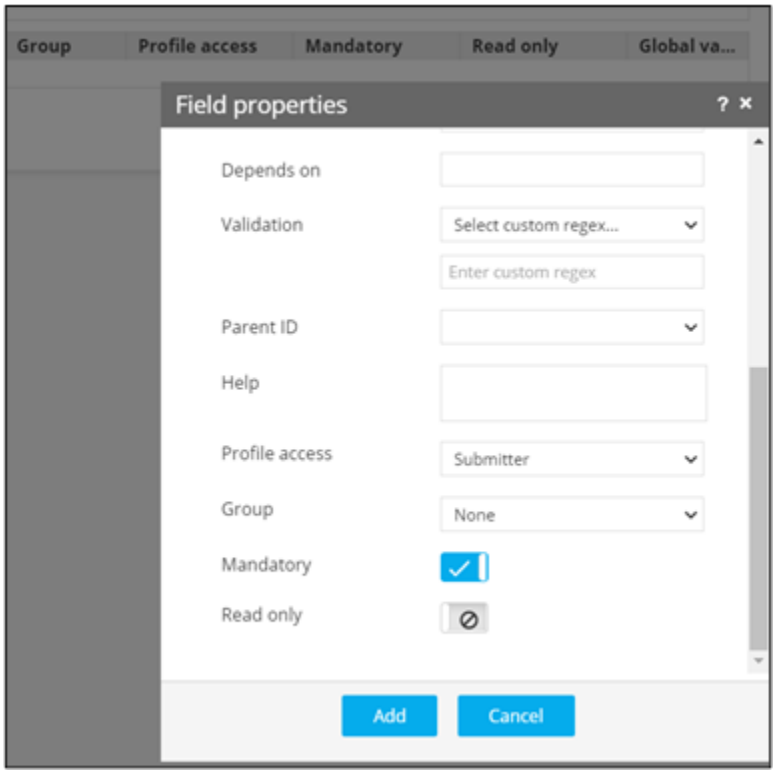
1. Design a workflow.
2. From the **User Interface** section, drag and drop the **Form** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. To design a new form, in the **Gallery** window, click **(Create)** icon.
4. Under the **Form Builder** section, from the upper right corner of the screen, click **(Add)** icon.
5. In the **Field properties** window, enter or select the field information.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Field properties

Field	Description
***Label name**	• Enter a valid label or field name such as Username, Enter IP address, Environment Type, Environment Info and so on. • 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).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
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, and 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.
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.
** : Mandatory fields*	

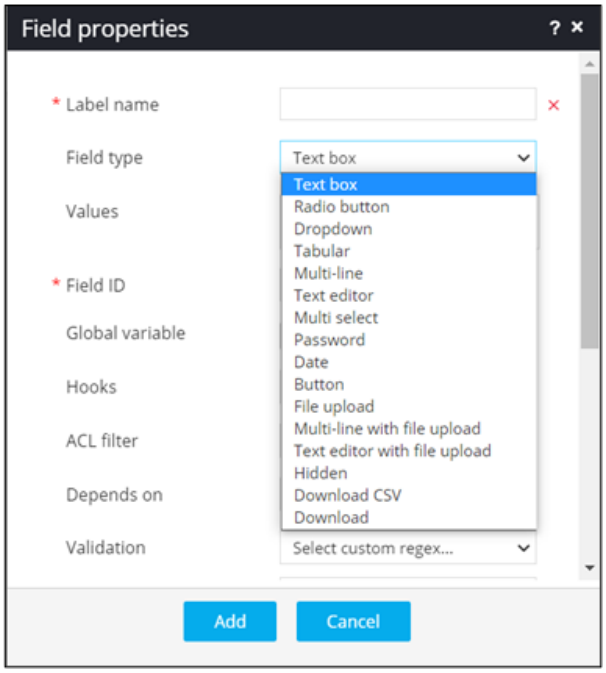
- Field Types
- Validating Form Fields
- Defining Dependencies between Form Fields

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.



- 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

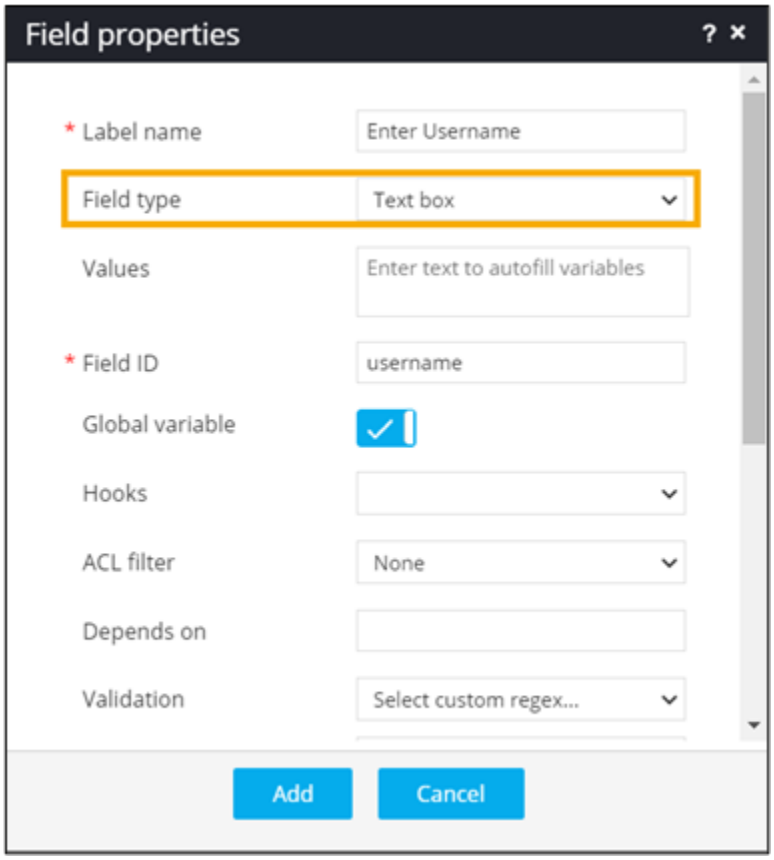
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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**, from the top right corner of the screen, click **(Add)** icon 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**.

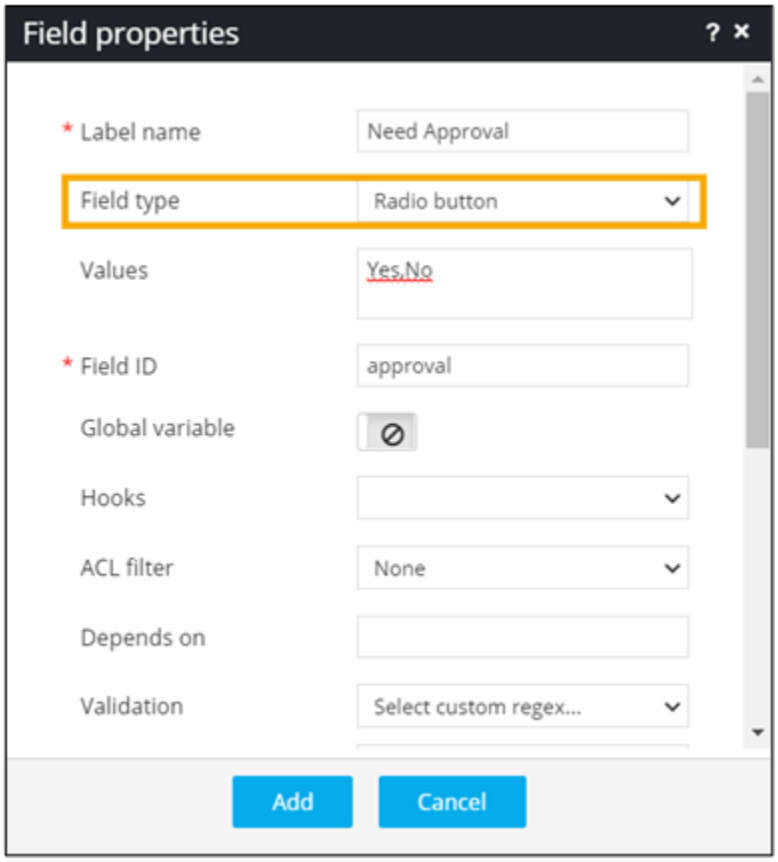
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. To see the text box added to the form, from the top right corner of the screen, click **(View)** icon.

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 **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the field information as shown.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

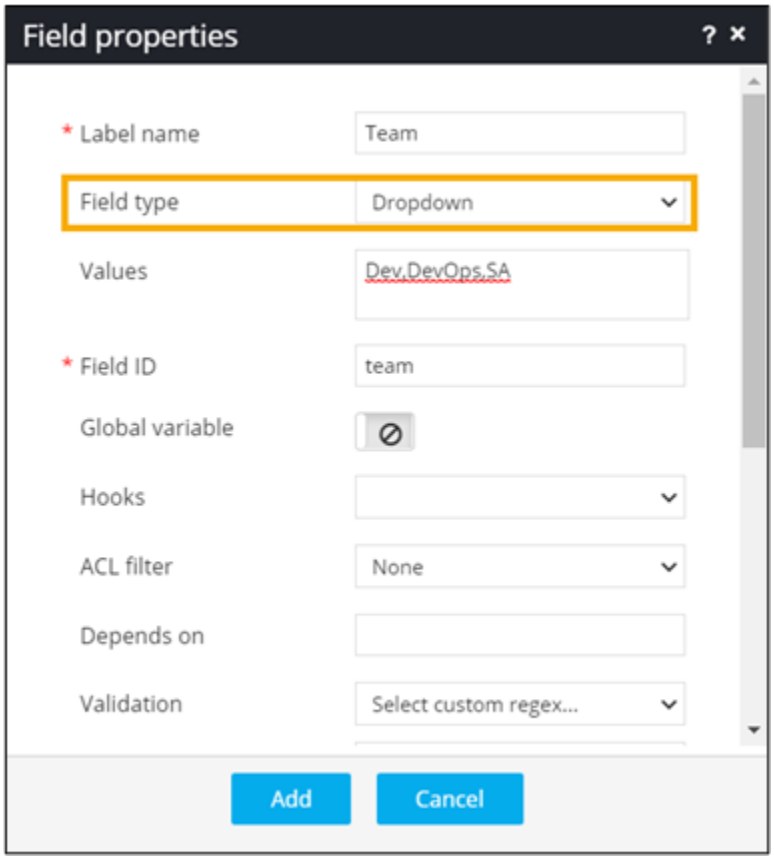
3. To add this field to the form, click **Add**.
4. To see the Radio button added to the form, from the top right corner of the screen, click **(View)** icon.

Field Type - Dropdown

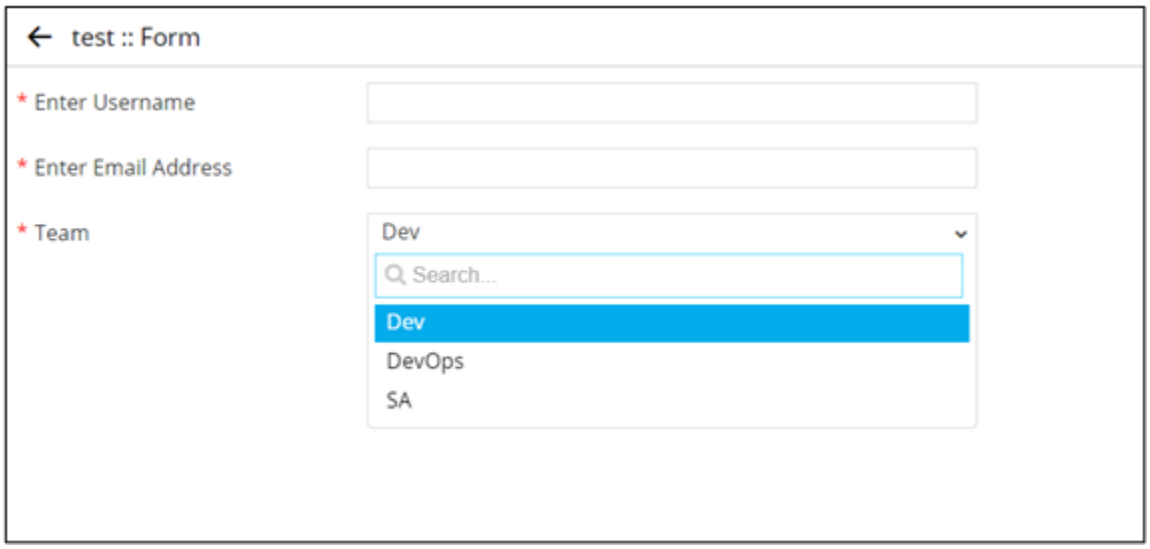
You can define a dropdown element with multiple values to select from in the form.

1. To add a **Dropdown** button to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the information as shown.

Copyright © 2025 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, from the top right corner of the screen, click **(View)** icon.

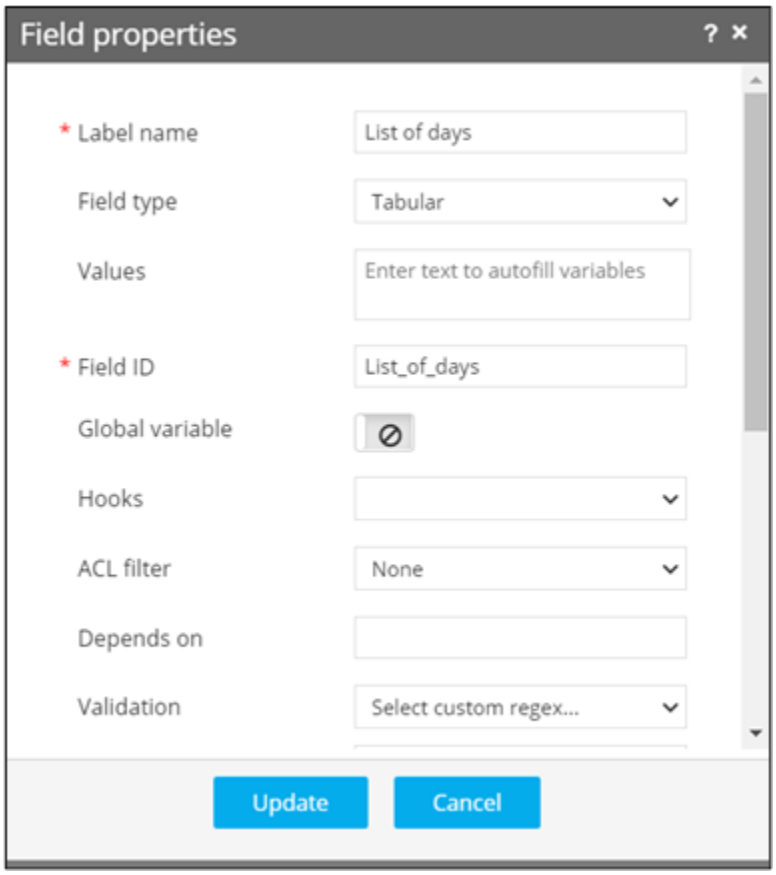


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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 **(Add)** icon in the command bar.
2. In the **Field properties** window, define a form field with field type as **Tabular**, with the respective Field ID.



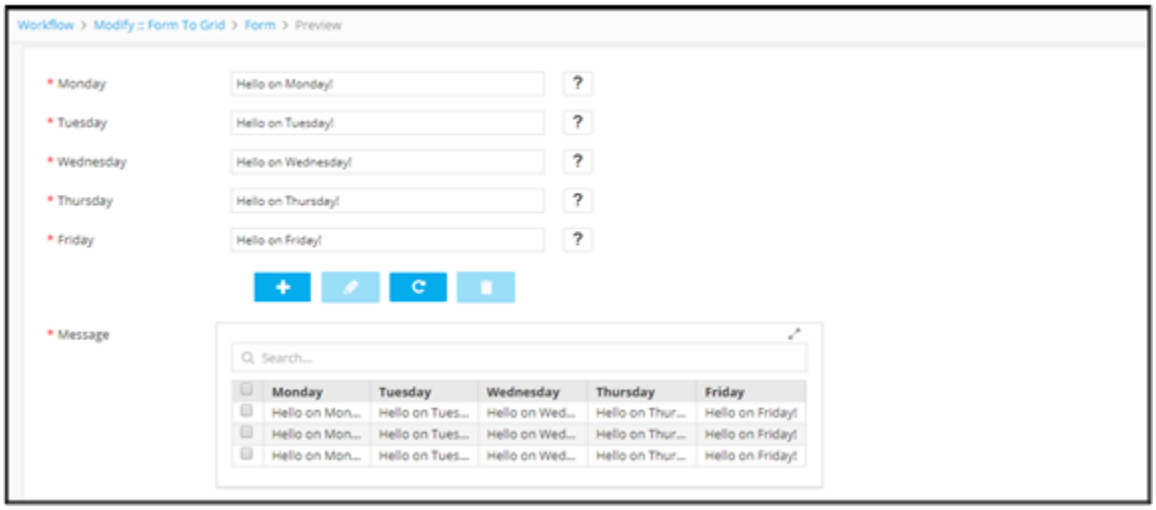
3. Define other form fields that are required for the tabular component.
   For example: Label name: Monday, Field type: Text box, Values: Hello on Monday.
4. Link the Parent ID (the field ID of the parent tabular element) against these fields.

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Label name	Field type	Values	Field ID	Parent ID
List of days	Tabular		List_of_days	

5. To see the Tabular field added to the form, click **(View)** icon.

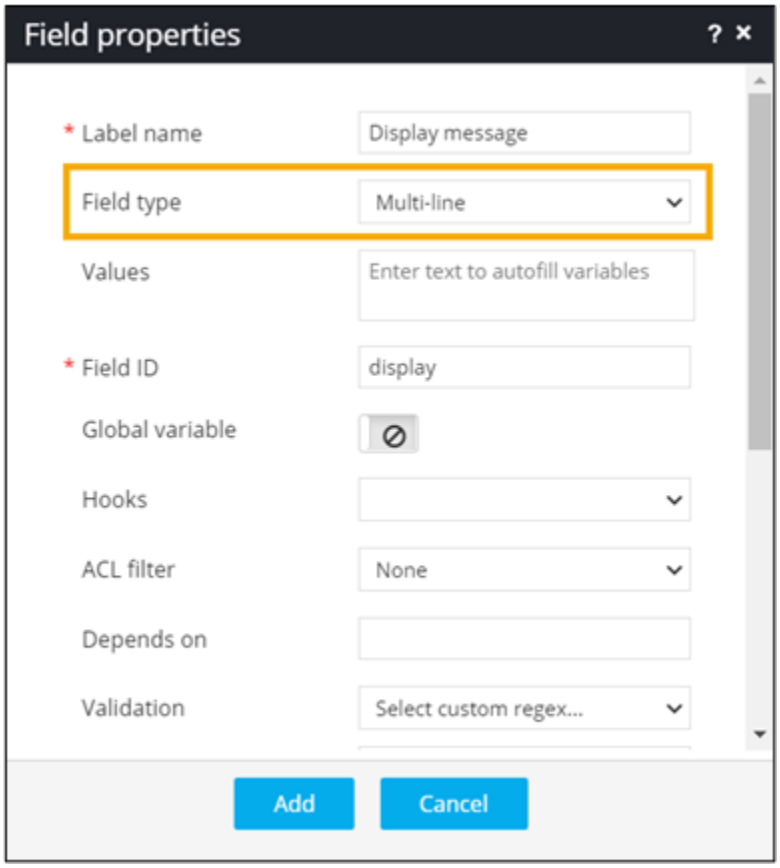


Field Type - Multi-line

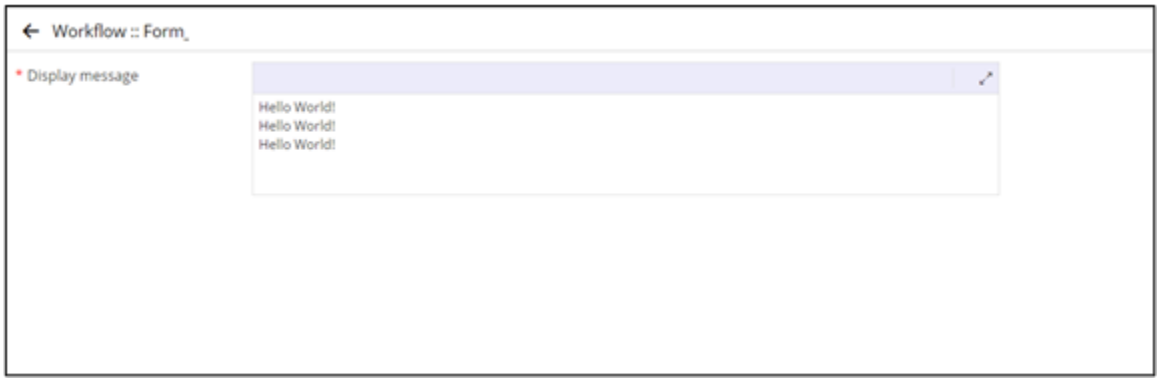
This field type in the form builder allows you to capture multiple lines of data within a form.

1. To add a **Multi-line** field to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the field information as shown.

Copyright © 2025 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, from the top right corner of the screen, click **(View)** icon.

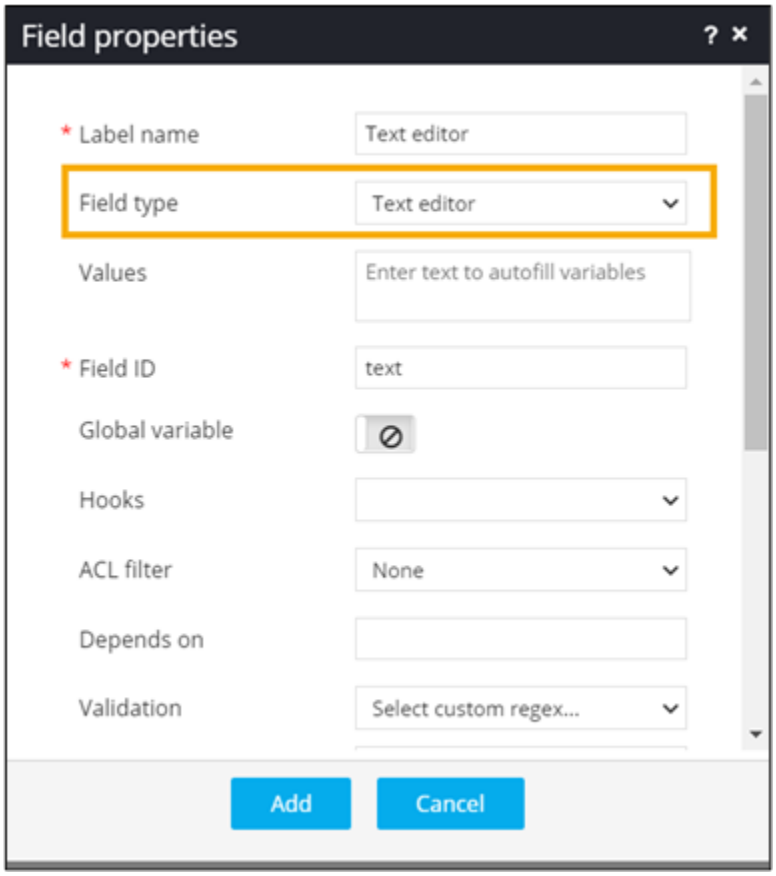


Field Type - Text editor

This field type allows you to add a text editor to their form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

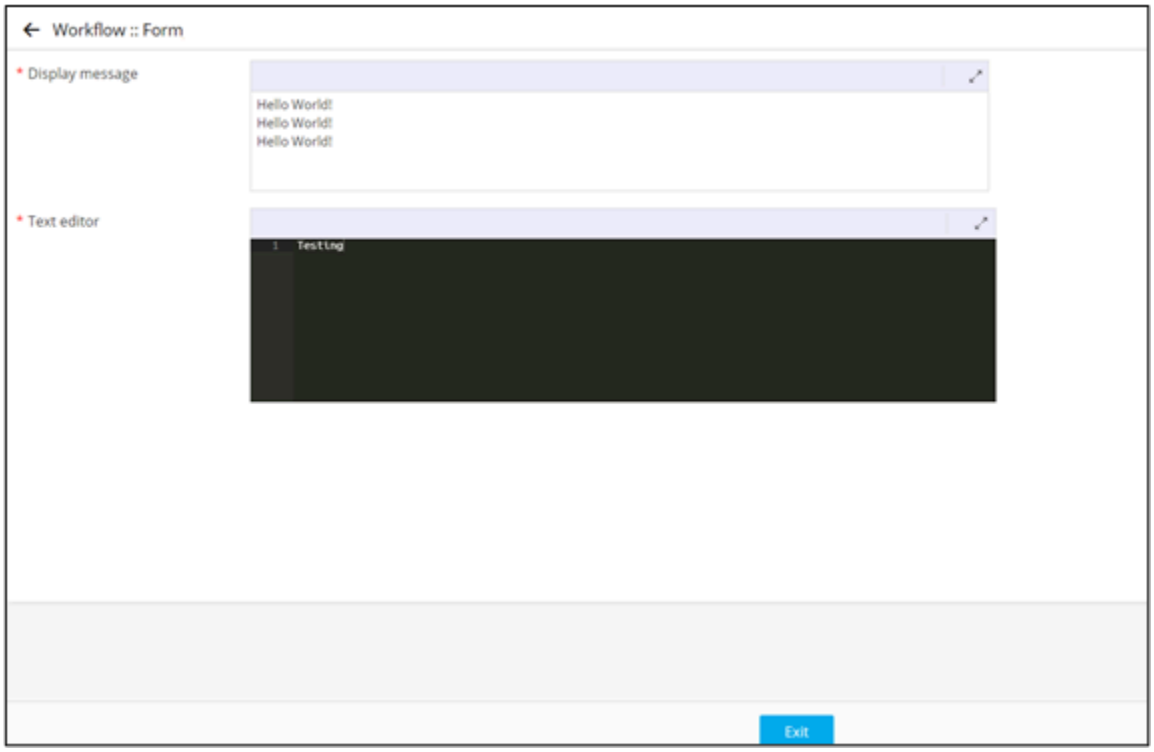
1. To add a **Text editor** field to the form, under **Form builder**, click + **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select information as shown.



3. To add the form field, click **Add**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. To see the field added to the form, click **(View)** icon.

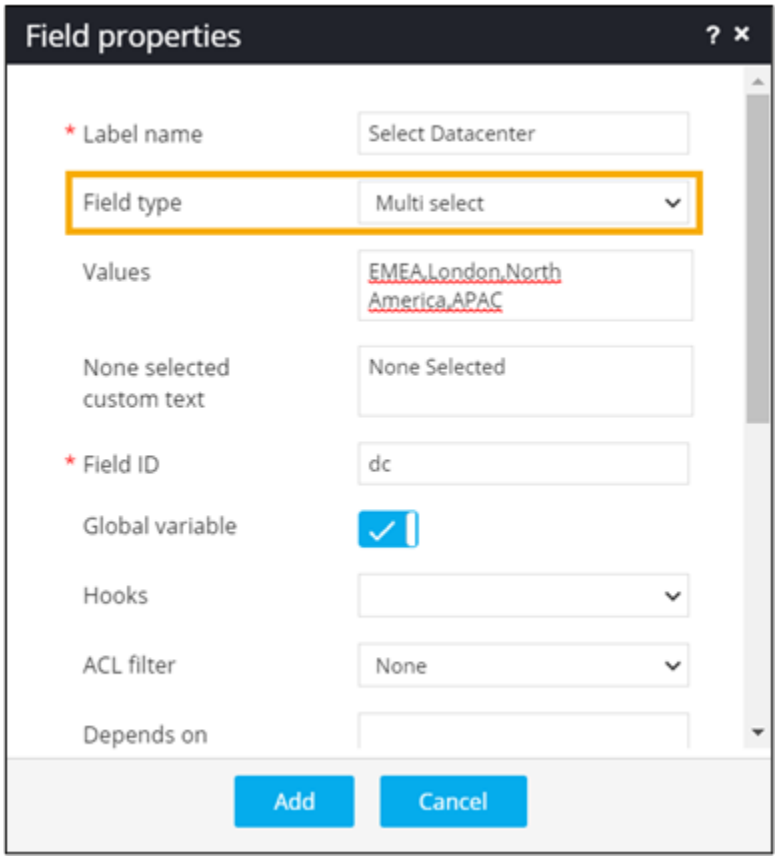


Field Type - Multi select

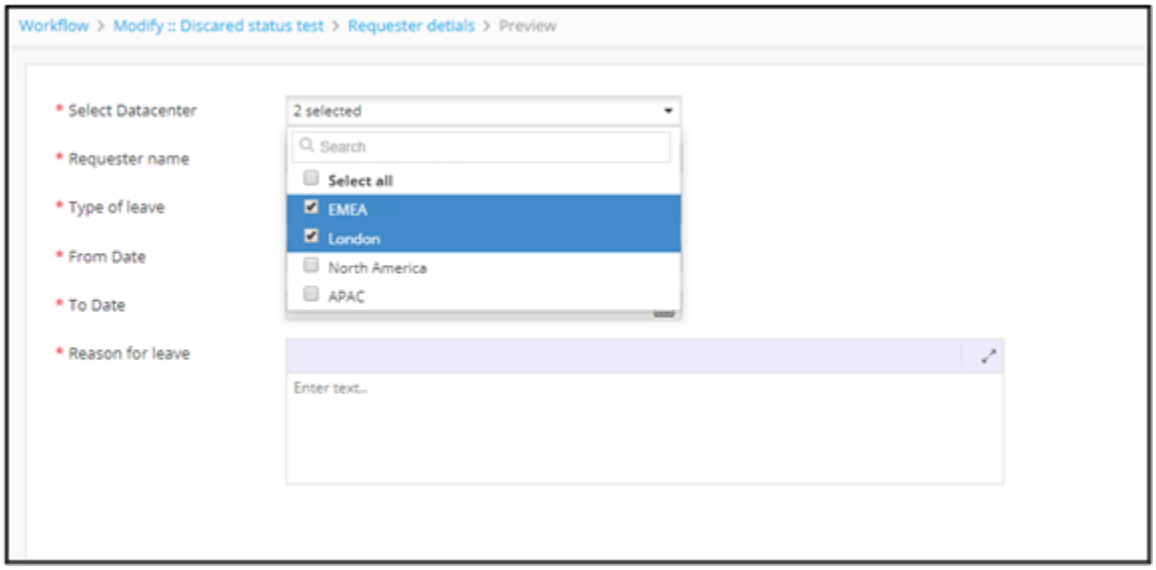
This field type in the form builder allows you to search and select multiple values from a specific dropdown form field within the request form.

1. To add a **Multi-select** field to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select field information as shown here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



3. To add this field to the form, click **Add**.
4. To see the field added to the form, from the top right corner of the screen, click **(View)** icon.

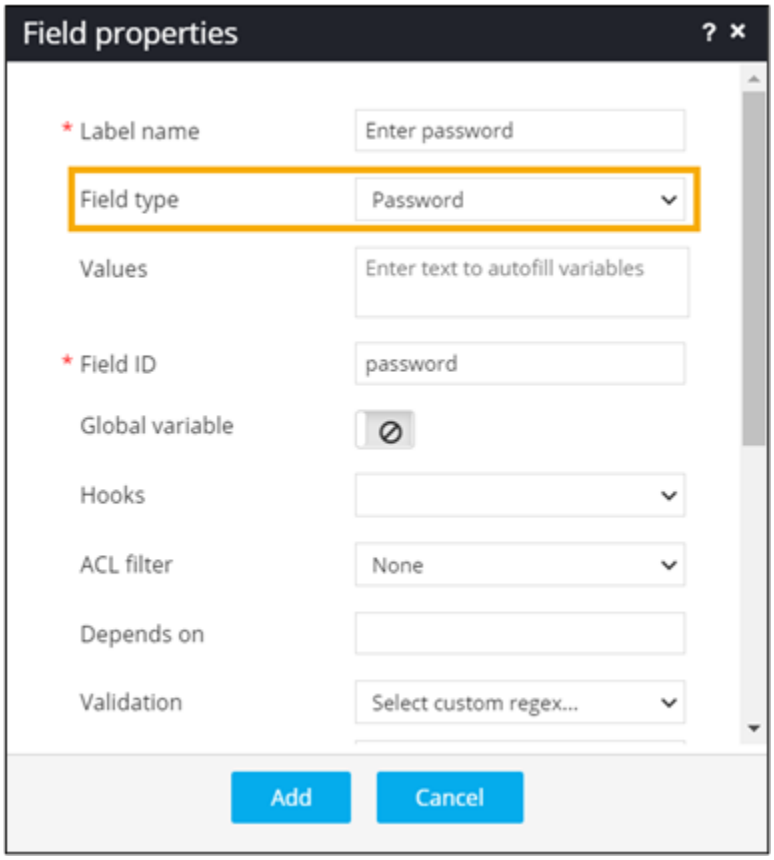


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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 **(Add)** icon 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**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

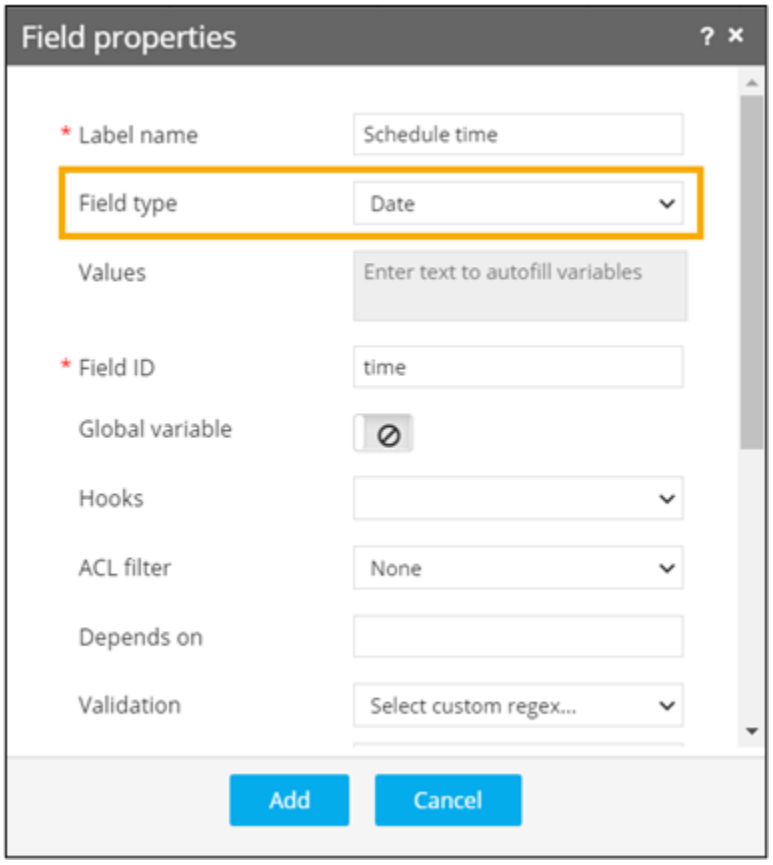
4. To see the field added to the form, from the top right corner of the screen, click **(View)** icon.

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 **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the field information as shown here.

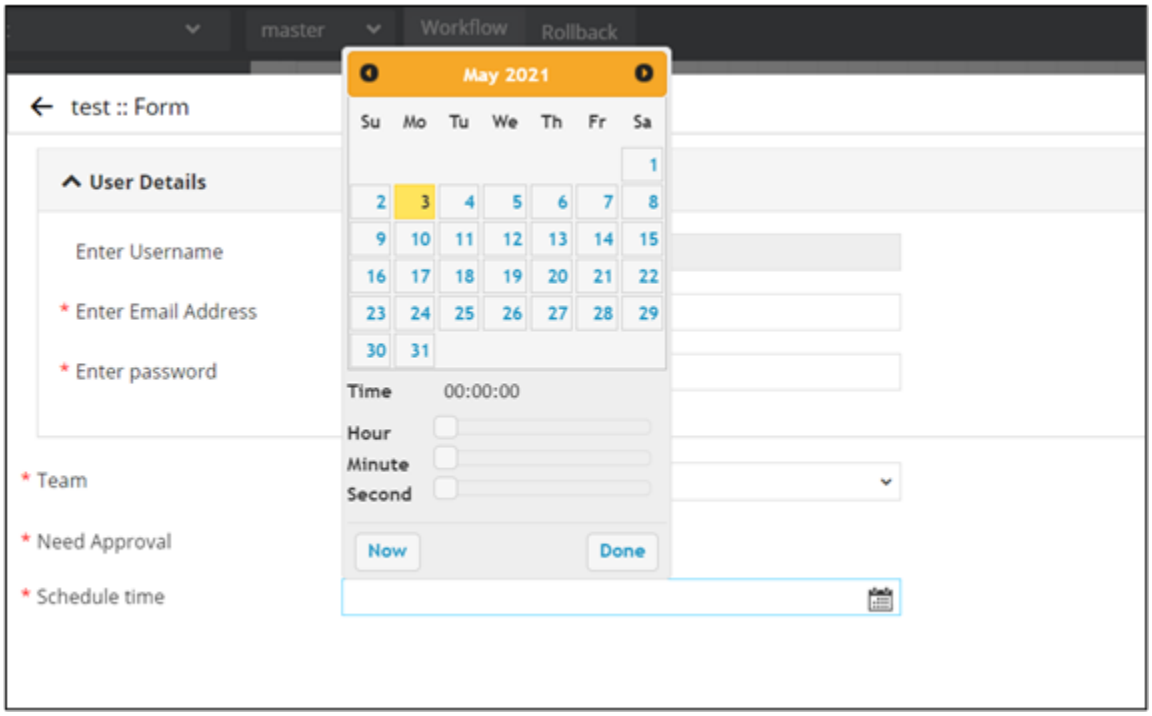
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



3. Click **Add**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. To see the field added to the form, click **(View)** icon.



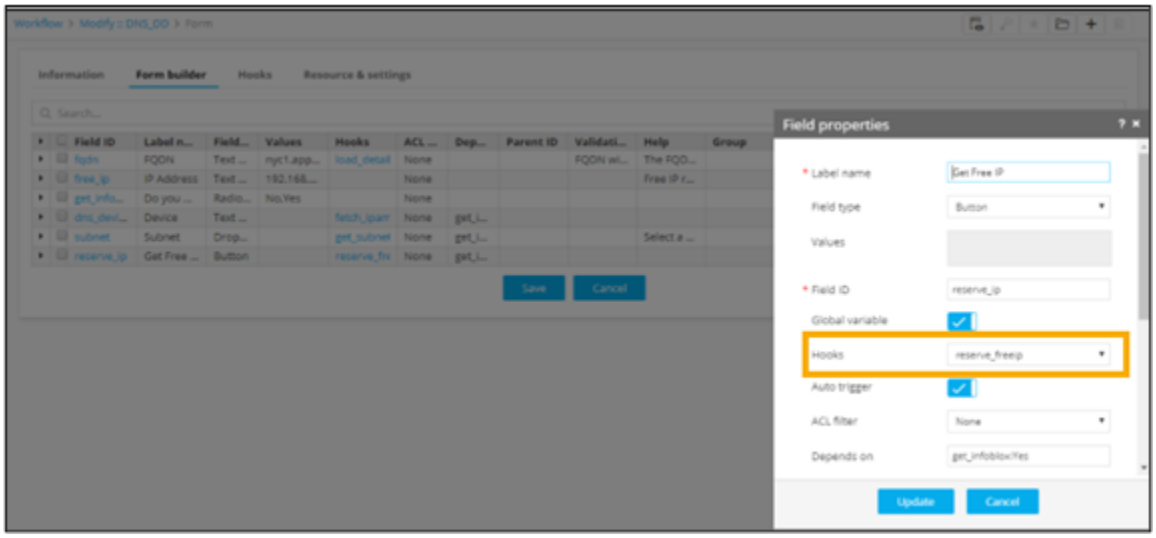
Field Type - Button

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.

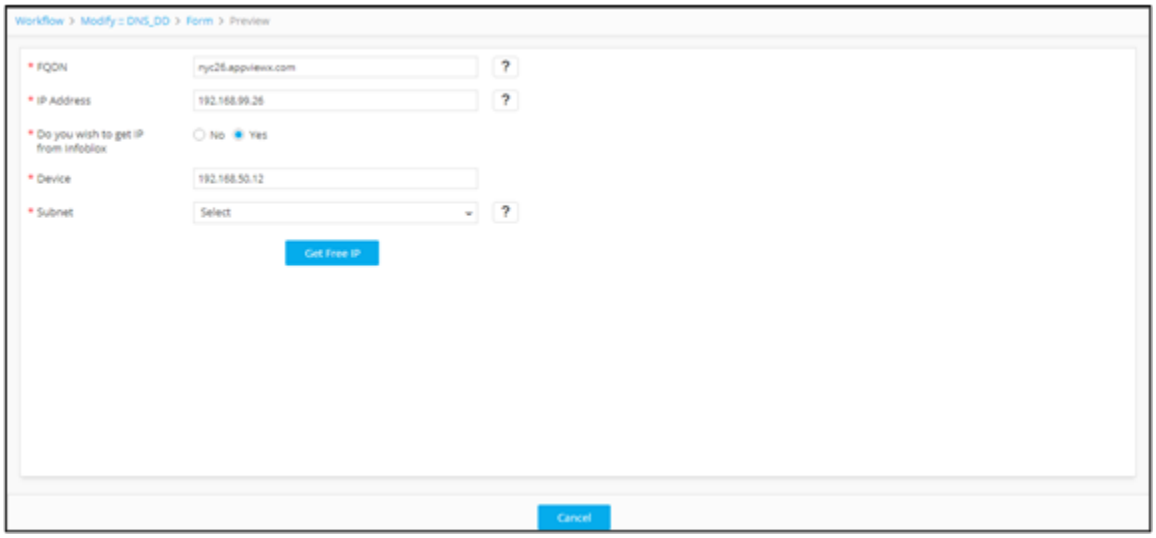
A relevant associate script must be mapped against the button field for it to perform a specific operation or redirect the output to a dependent field.

1. To add a **Button** to the form, under **Form builder**, click **(Add)** icon 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 © 2025 AppViewX, Inc. All Rights Reserved.



4. To see the button added to the form, from the top right corner of the screen, click **(View)** icon.

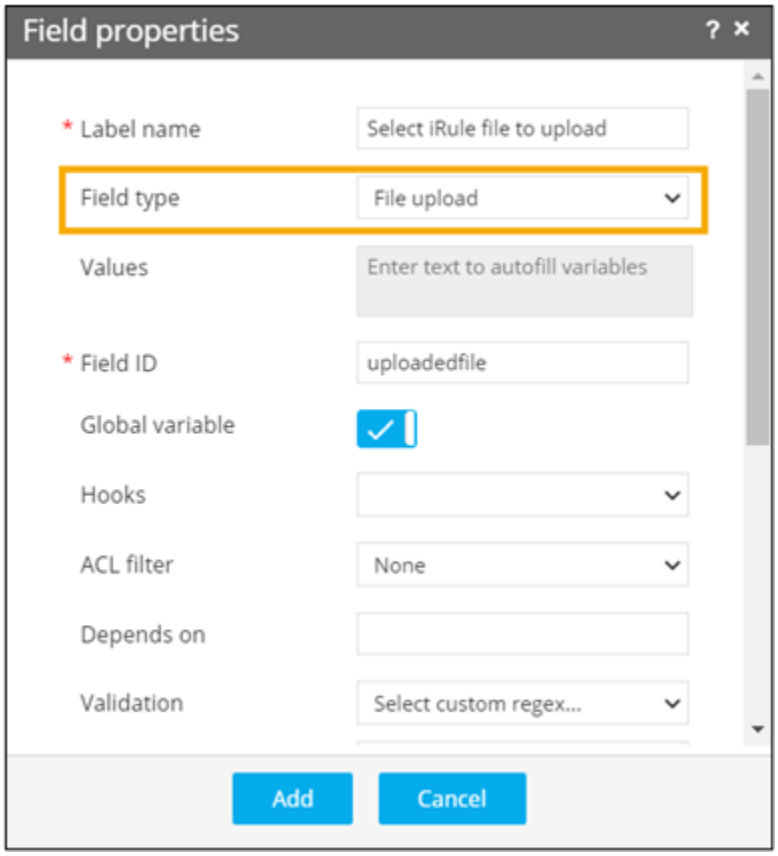


Field Type - File upload

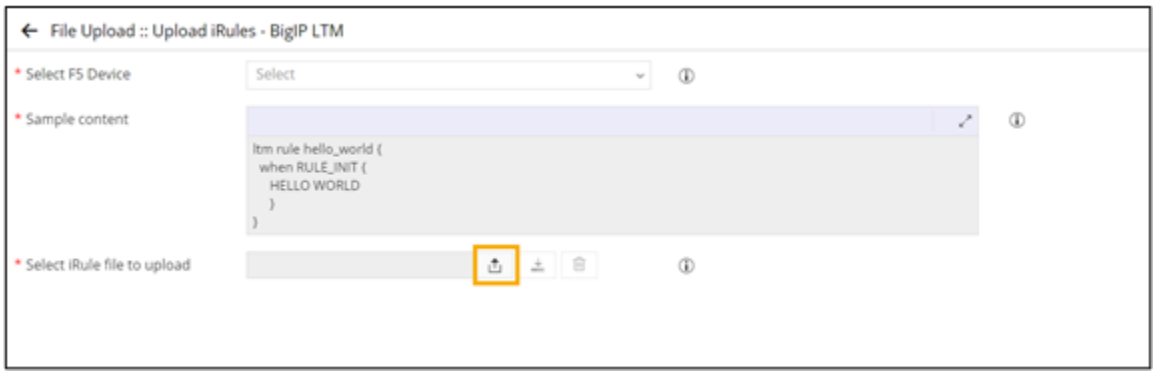
This form field allows you to add a file upload element to your form. You can upload files of maximum 2 MB in .txt, .pem, .crt, .csv formats.

1. To add a **File Upload** field to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the field information as shown here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



3. To add this field to the form, click **Add**.
4. To see the field added to the form, from the top right corner of the screen, click **(View)** icon.

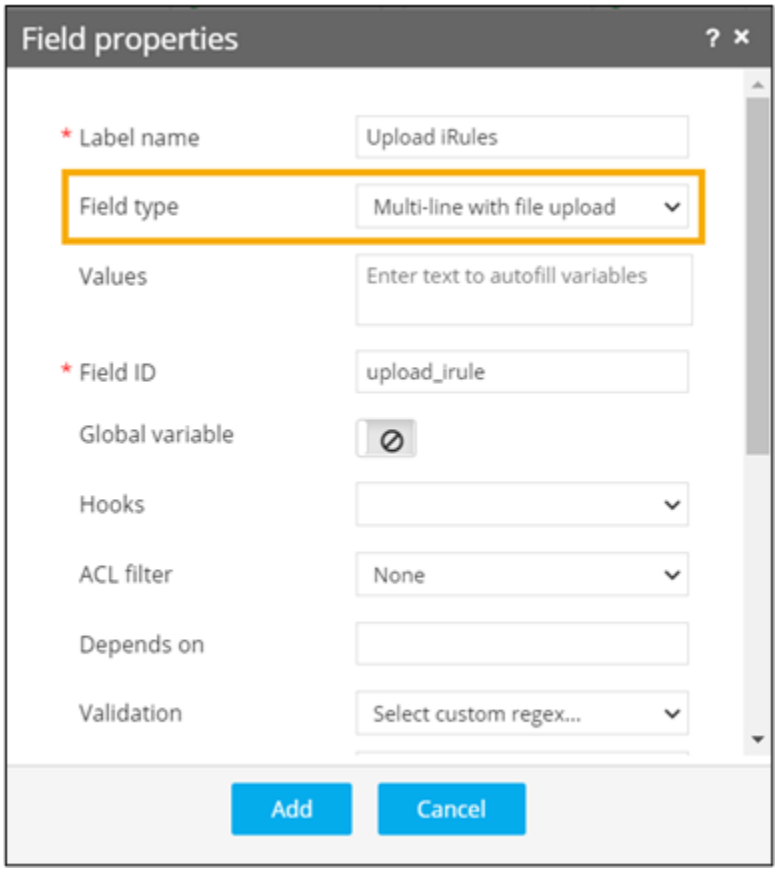


Field Type - Multi-line with file upload

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

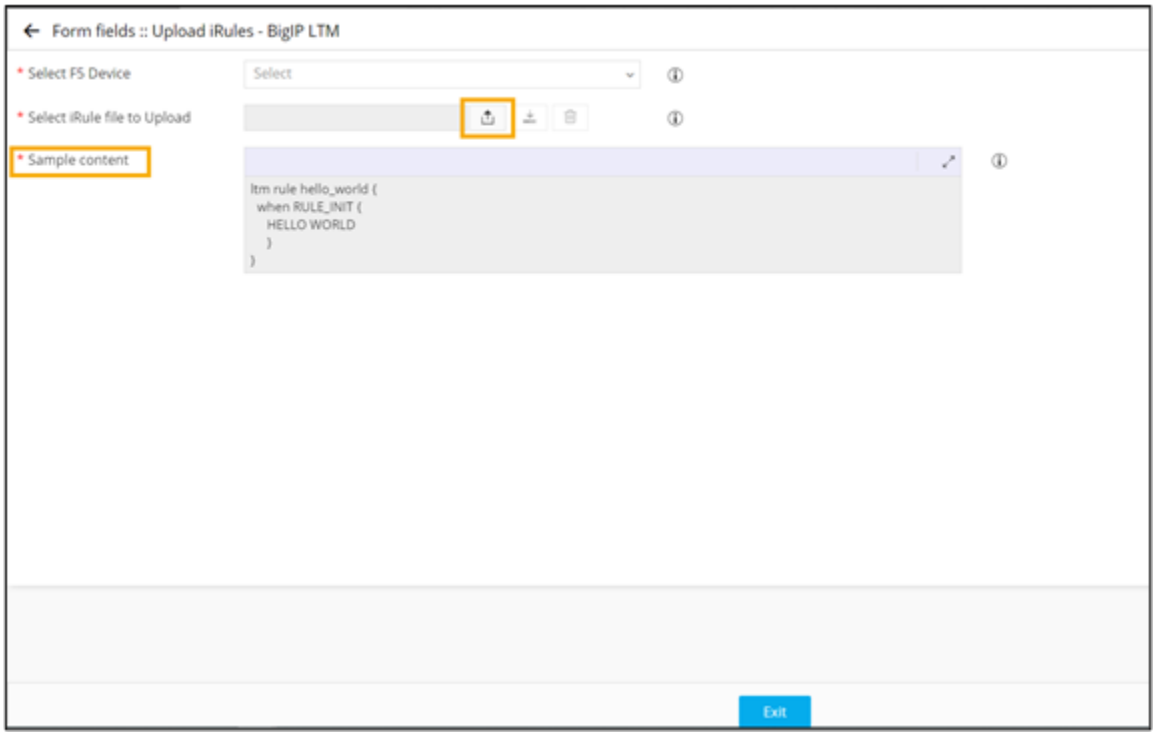
1. To add a **Multi-line with file upload** field to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the field information as shown.



3. To add this field to the form, click **Add**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. To see the field added to the form, from the top right corner of the screen, click **(View)** icon.



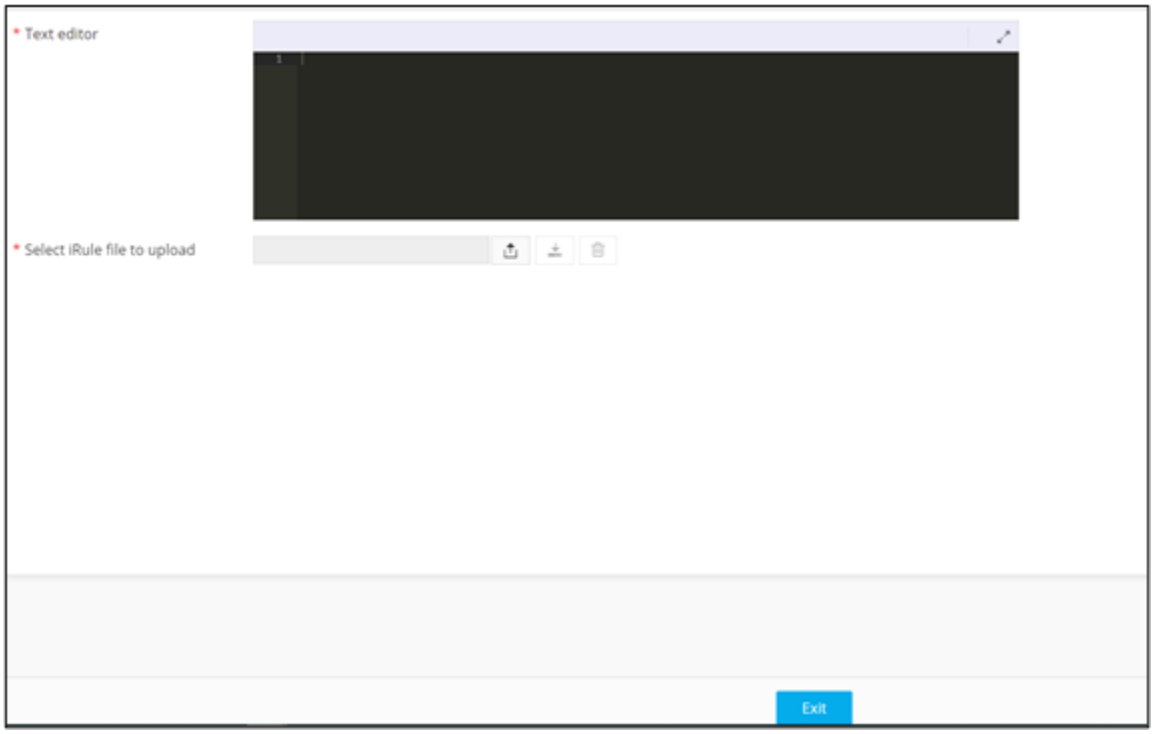
Field Type - Text editor with file upload

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 **(Add)** icon in the command bar.
2. In the **Field properties** window, add the **Text Editor** form field.
3. To add the form field, click **Add**.
4. Add the **File upload** form field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

5. To see the field added to the form, click **(View)** icon.

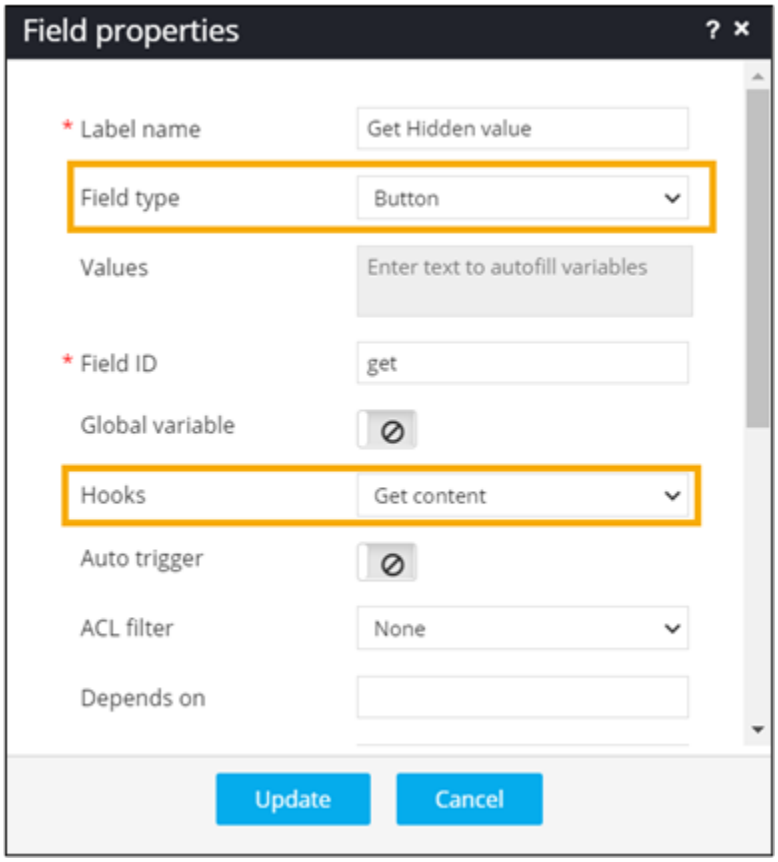


Field Type - Hidden

You can use the Hidden field to hide certain values or content in the form and fetch it only when required.

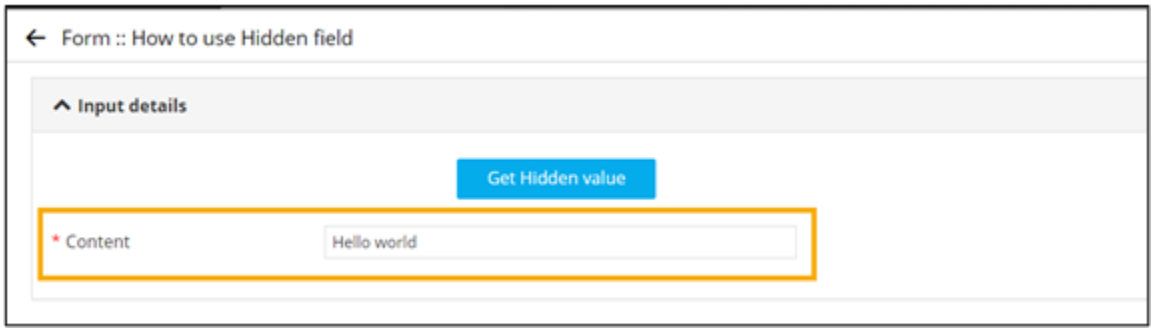
1. To add a **Hidden** field to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter/select the field information as shown here.
3. To fetch hidden data, add the **Button** field and define the relevant hook.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. Add a **Text box** field.
5. To see the field added to the form, click **(View)** icon.
6. In the form preview, click **Get Hidden value**.

   Hidden value is displayed.

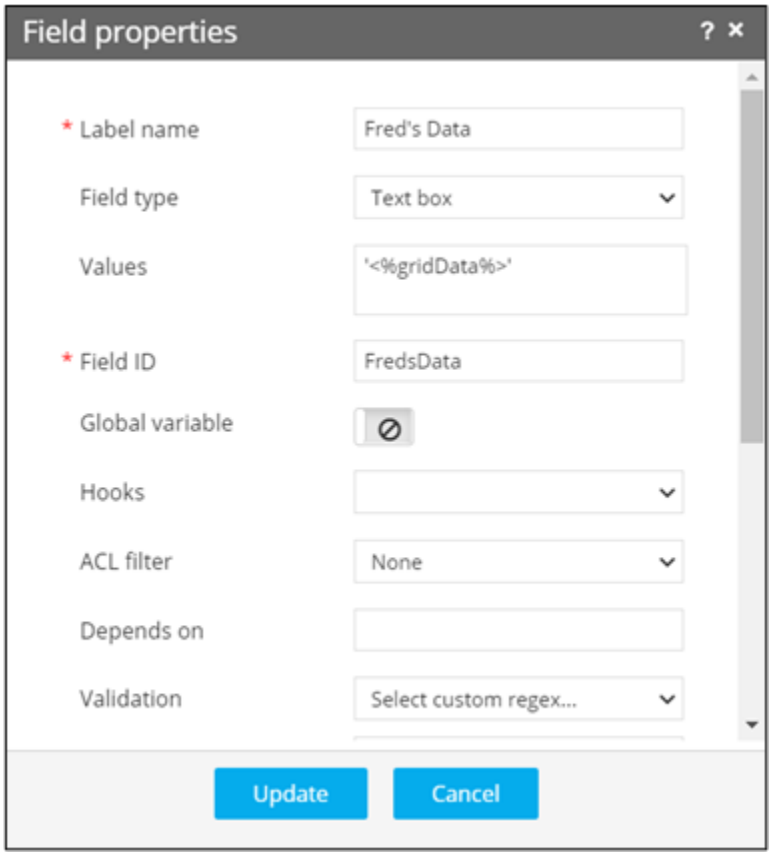


Field Type - Download CSV

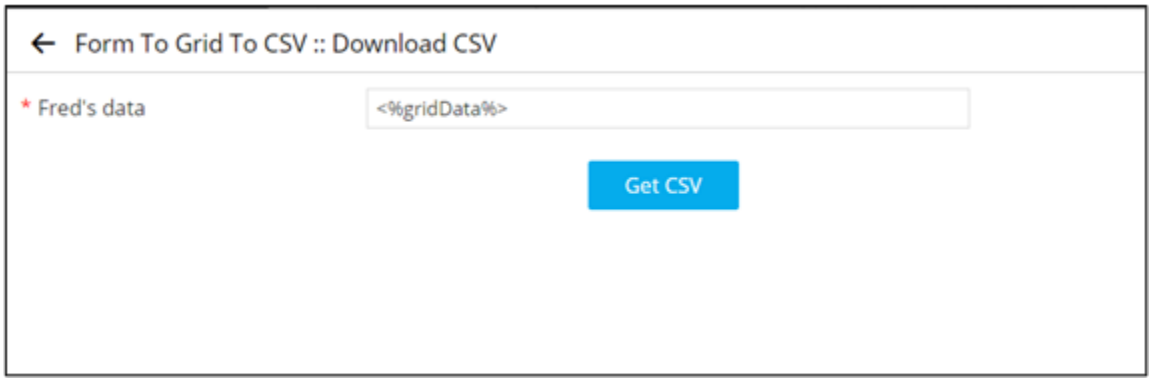
This form field allows you to download the data displayed in a grid to your machine as a .csv file.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. To add a **Download CSV** button to the form, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, enter or select the information as shown.



3. Add another form field (Get CSV) with field type as **Button**.
4. To see the field added to the form, click **(View)** icon.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field Type - Download

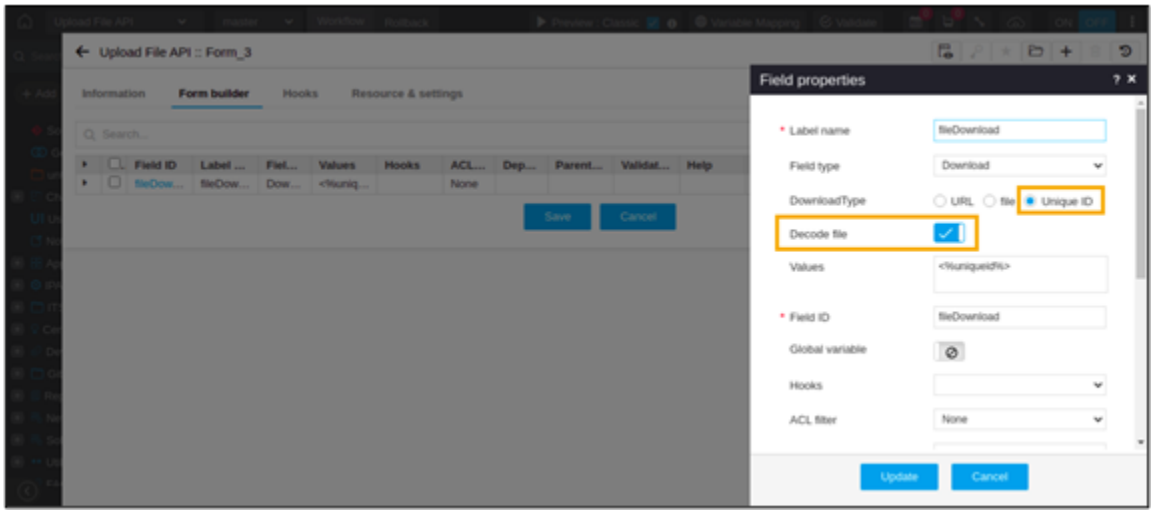
This form field allows you to download the data from the file to your machine.

- 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, under **Form builder**, click **(Add)** icon in the command bar.
2. In the **Field properties** window, add a **Download** field.
   The **Download Type** field displays three options:
   - **URL**
   - **File**
   - **Unique ID**.
3. 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.

4. To enable the **Decode file** option, turn on the toggle.

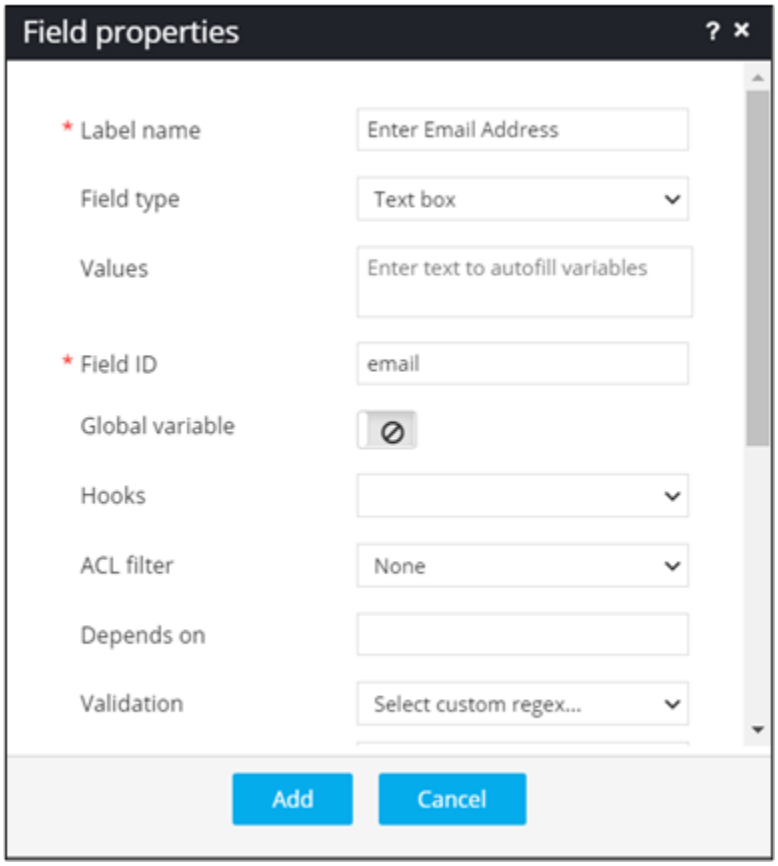


Validating Form Fields

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 © 2025 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 **(Add)** icon in the command bar.
4. In the **Field properties** window, enter or select the field information.



5. In the Validation field, select **Email**.

> **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, from the top right corner of the screen, click **(View)** icon.
   Only values that match the defined regex are accepted.

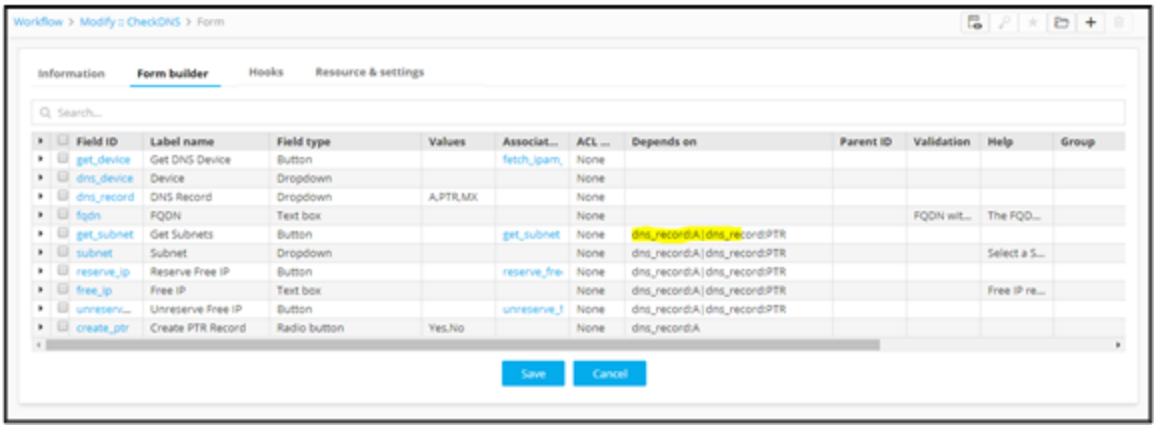
> **Note:** A is displayed next to the form field if the entered value does not match the defined regex.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.

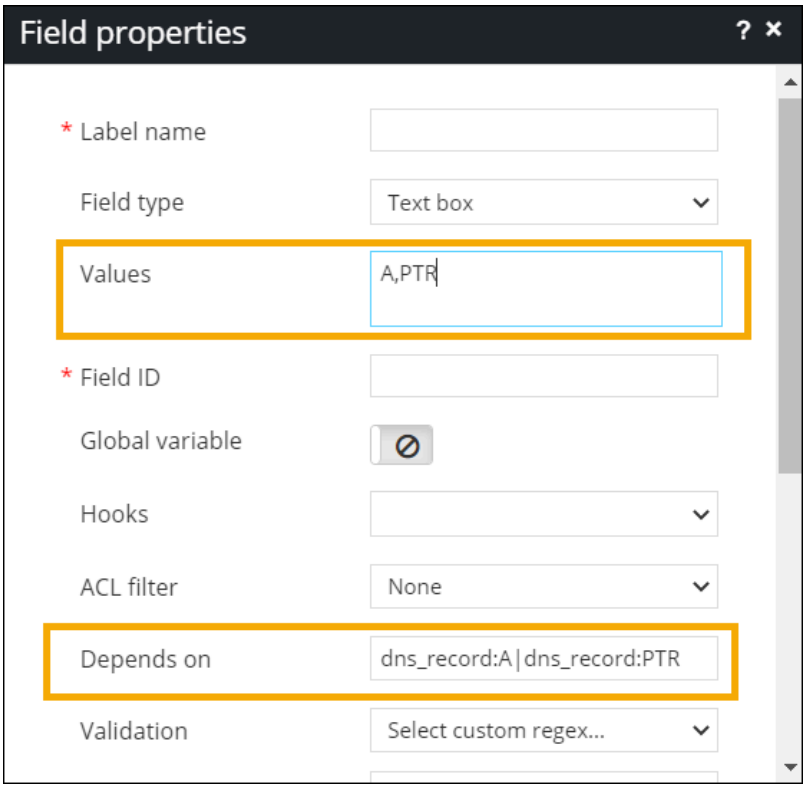
- Operators supported: ‘&’ (AND); and ‘|’ (OR)
- Defining dependency based on single field and value: [Field ID: Value]
- Defining dependency based on multiple fields and values: [Field ID1: Value1 & Field ID2: Value1], [Field ID1: Value1 | Field ID2: Value1]



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 **(Add)** icon 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.
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 © 2025 AppViewX, Inc. All Rights Reserved.

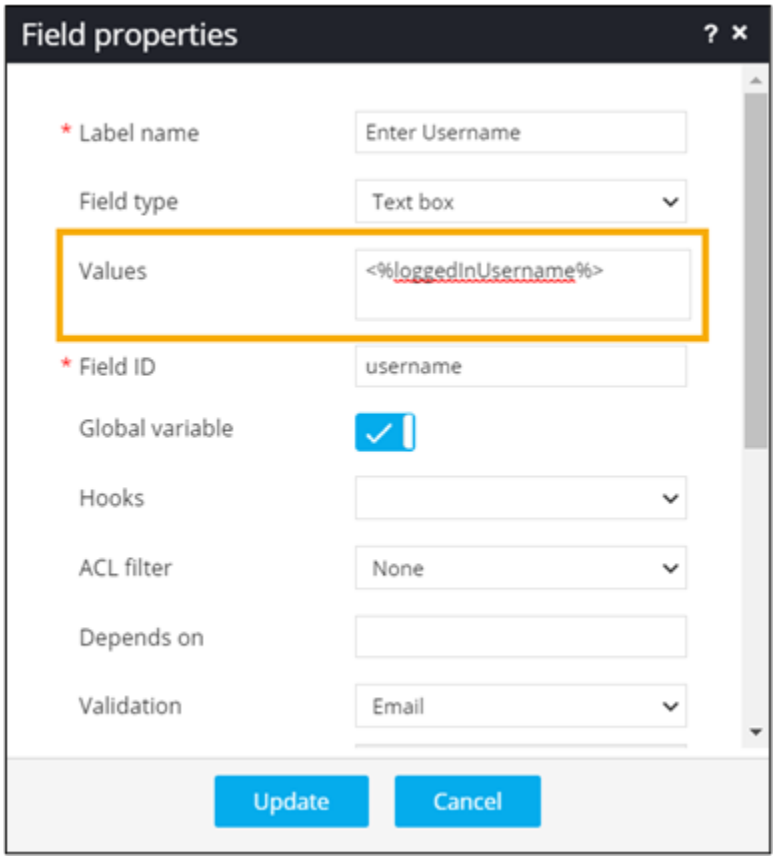


6. Add other form fields and define dependencies as required.
7. To see the form fields added to the form, from the top right corner of the screen, click **(View)** icon.

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 **(Add)** icon in the command bar.
4. In the **Field properties** window, enter or select the field information to define the form field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



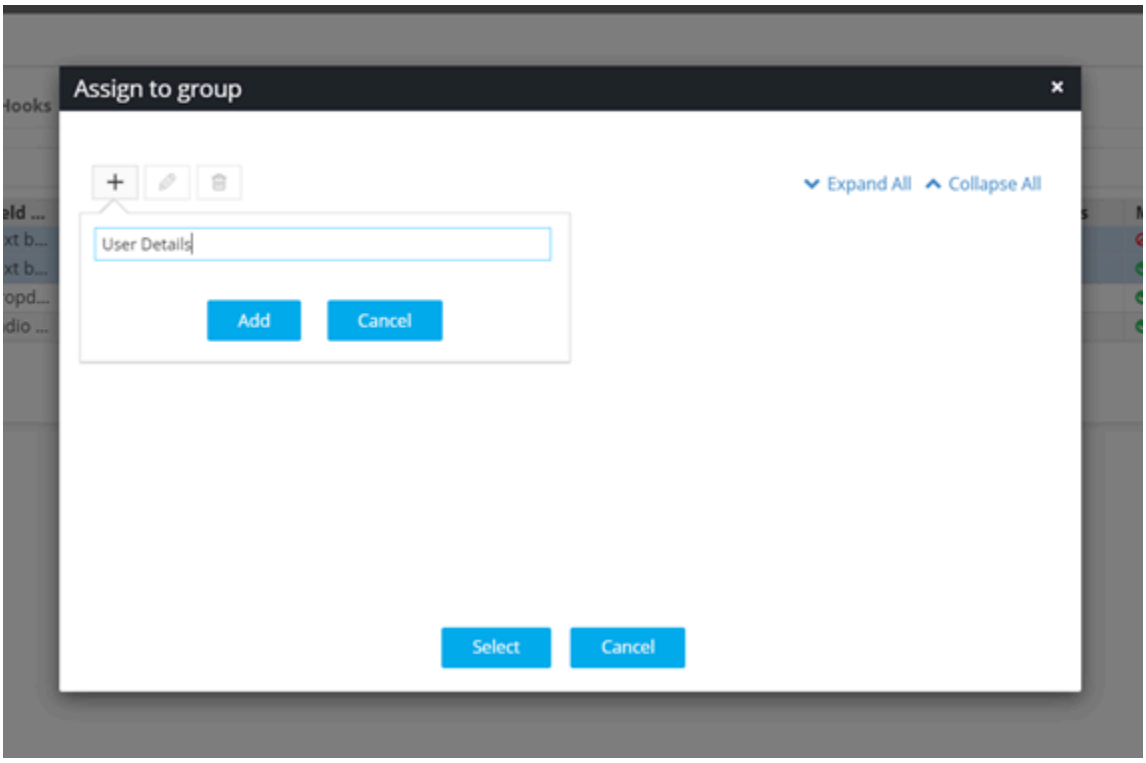
5. To define the field as **Read only**, turn on the toggle.
6. To see the Read only username field added to the form, from the top right corner of the screen, click **(View)** icon.

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, from the top right corner of the screen, click **(Folder)** icon.
5. To create a new group, in the **Assign to group** window, click **(Add)** icon and enter the group name.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

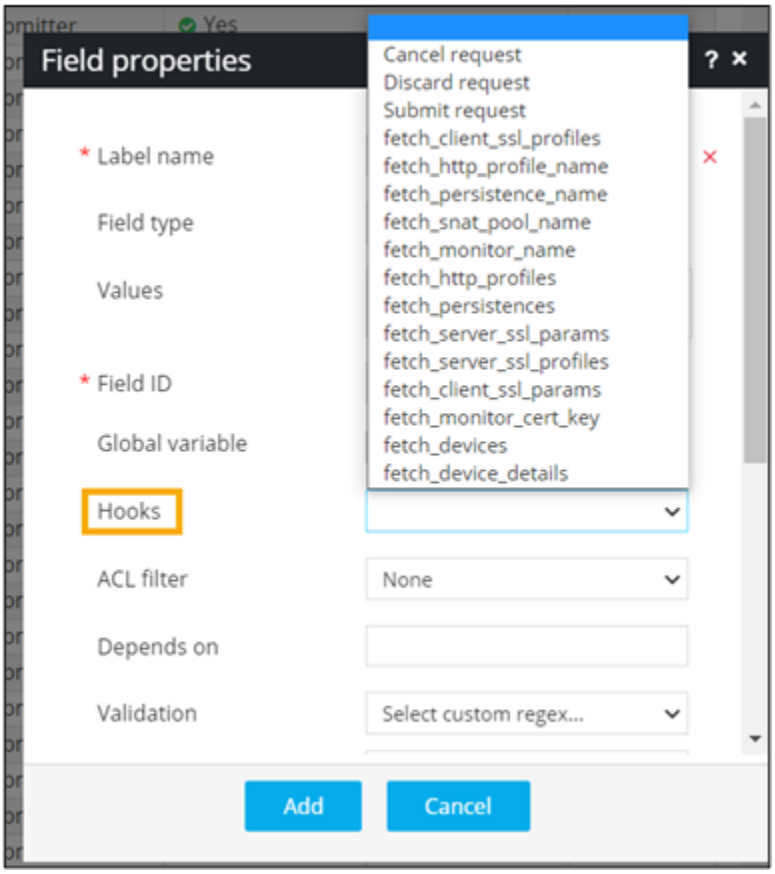


6. To see the form fields listed under the assigned group, from the top right corner of the scree, click **(View)** icon.

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 © 2025 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**.

- Using Scripts with Forms
- Auto-trigger Scripts
- Default Script for Cancel and Discard Actions
- Define Custom Notification Messages using Scripts
- Defining Query Explorer to Build Forms

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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:

1. Design a form using the following form fields:

Field Name	Field ID	Type
Get type of users	get	Button
User type	type	Dropdown

2. Define an associate script to get the **User type** and map the script to field **Get types of users**.

Script	Description
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.

3. From the menu on the top, click **(View)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Clicking the **Get type of users** button triggers the script and populated the **User type** field.

Triggering Script to Get Device List

To query the database by defining an associate script and get a list of F5 devices:

1. Design a form using the following form fields:

Field Name	Field ID	Type
Get device list	get	Button
Select the device	device	Dropdown

2. Define an associate script to get the **devices** and map the script to field **Get**.

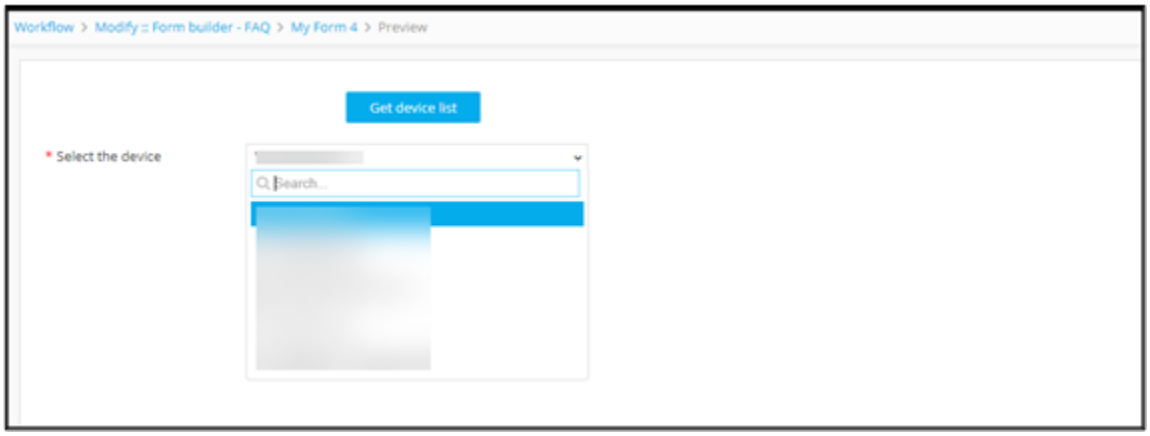
Script	Description
### To make use of the helper script ### *sys.path.insert(0,AVX::HELPER)* *sys.path.insert(0,AVX::DEPENDENCIES)* ### Importing the appviewx helper ### import appviewx ### calling the inbuilt function the helper script to connect with the DB ### `db_connection = appviewx.db_connection() device_collection = db_connection.appviewx.device`	Where **device** is the Field ID to which the retrieved devices are mapped to on the form

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Script	Description

###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))`	

3. From the menu on the top, click **(View)** icon.



Clicking the **Get device list** button triggers the script and populates the **Select the device** field.

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. Design a form using the following form fields:

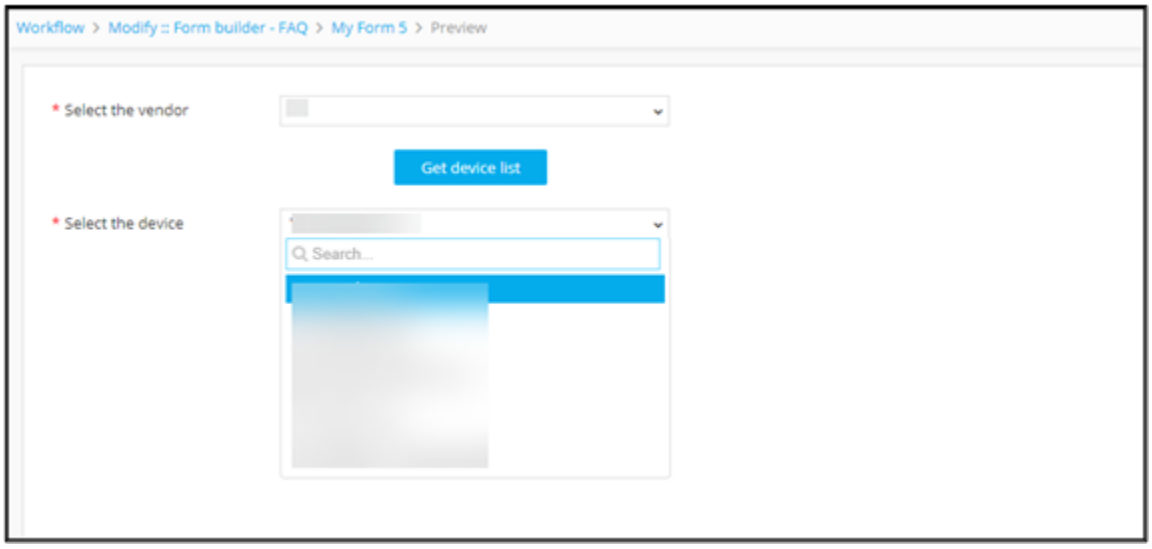
Field Name	Field ID	Type
Select vendor	type	Dropdown
Get device list	get	Button
Select the device	device	Dropdown

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

2. Define an associate script to get the **devices** and map the script to field **Get**.

Script	Description
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.

3. From the menu on the top, click **(View)** icon.
   The script is triggered to get the device list by vendor.

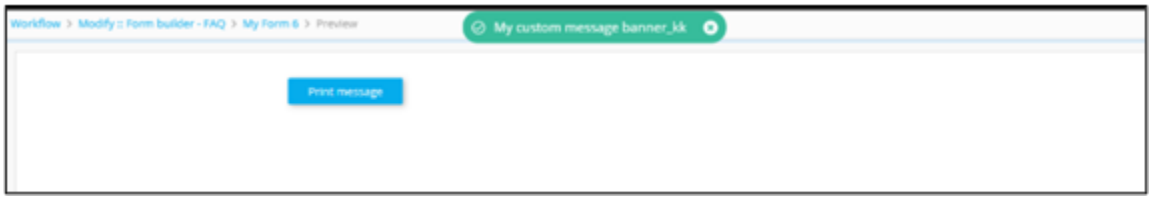


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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"}))`
3. From the menu on the top, click **(View)** icon.
   Clicking the **Print message** button triggers the script to display the custom message.



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. Design a form using the following form fields:

Field Name	Field ID	Type
Select device	device	Dropdown
Enter Virtual Name	name	Text Box
Get Virtual Details	get_virt	Button

2. Define an associate script and script logic.
3. Map the associate script against the relevant form field ID.
   Associate Script logic

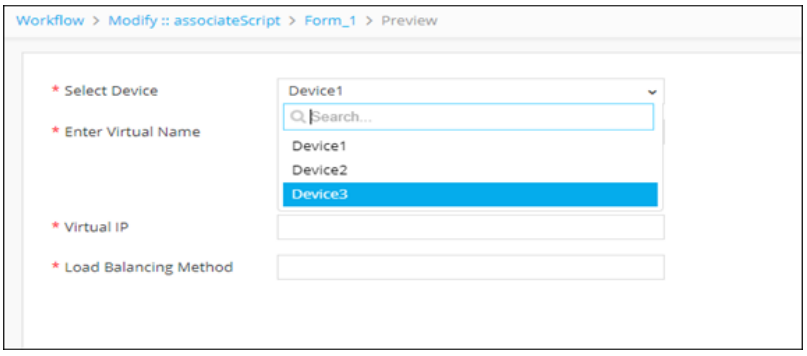
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
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()
```

4. Define two more form fields to show the Virtual IP and Load balancing method.

Field Name	Field ID	Type
Virtual IP	virtual_ip	Text Box
Load Balancing Method	lb_method	Text Box

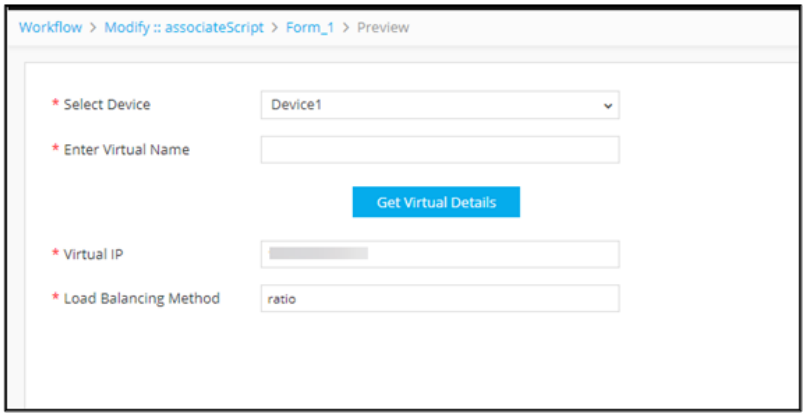
5. From the menu on the top, click **(View)** icon.
6. In the **Select Device** field, select the device as **Device1**.



7. Enter the virtual name and click the **Get Virtual Details** button.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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:

1. Design a form using the following form fields:

Field Name	Field ID	Type
Get F5 LTM Device List	DeviceList	Button
F5 LTM Device	Device	Dropdown

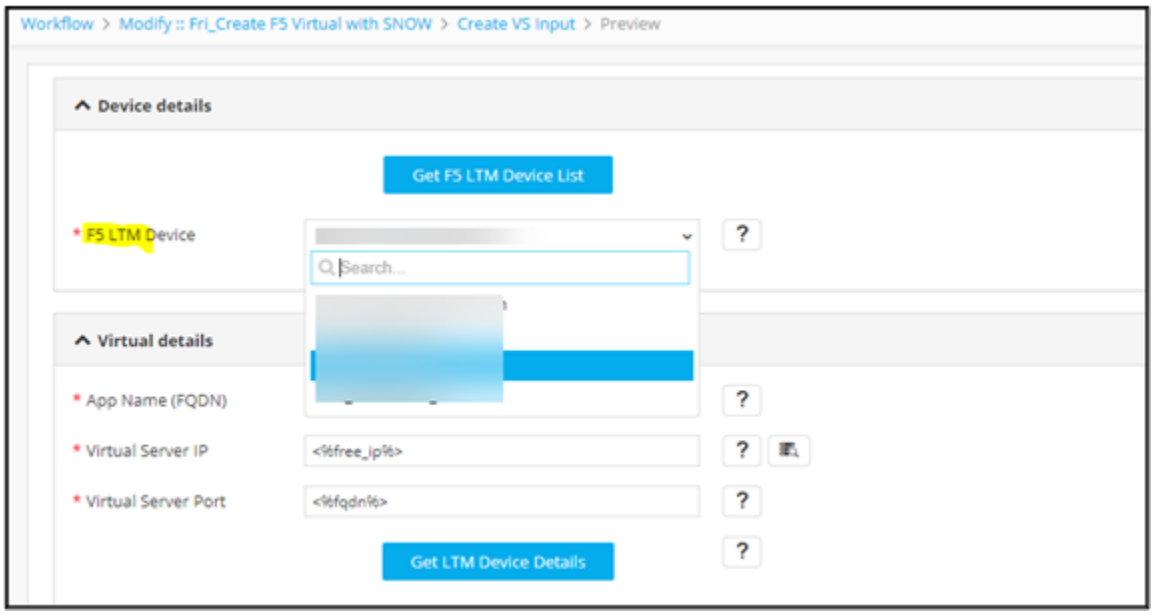
2. Define custom script logic. Use ‘Ctrl + Space’ in the script editor to list all the pre-defined helper scripts and embed the appropriate helper script(s) within an ‘associate script’.
   Associate Script

Copyright © 2025 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

3. From the menu on the top, click **(View)** icon.
4. To execute the associate script on the form, click **Get F5 LTM Device List**.

   The **F5 LTM Device** field is populated with the device names based on the script.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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:

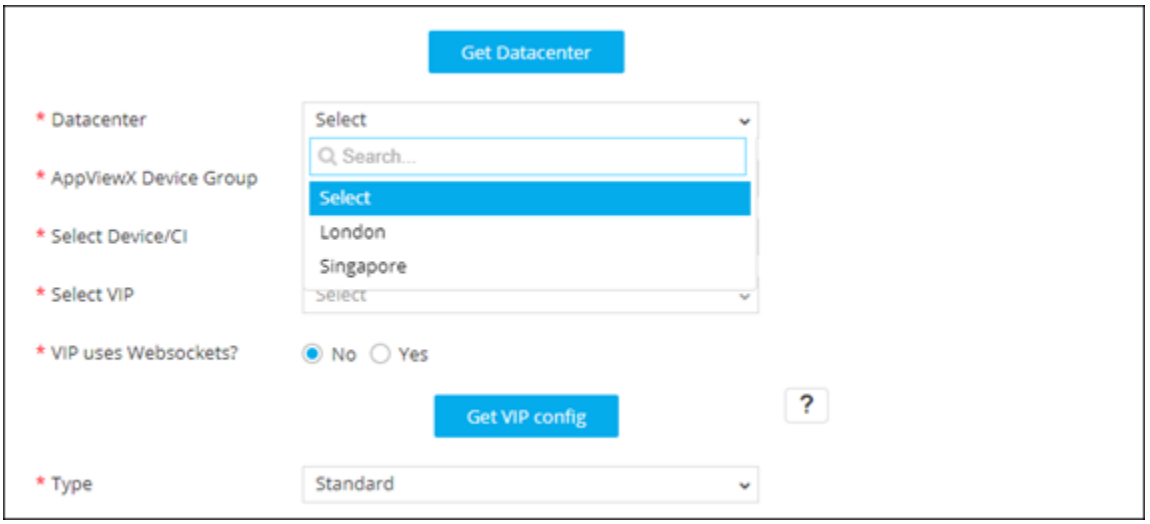
- Auto trigger works for the first form field on loading.
- The input from the form must be from a radio button or dropdown form field.
- Based on the input, auto trigger gets executed ONLY for the next immediate field.

1. Design a form using the following fields and associate scripts:

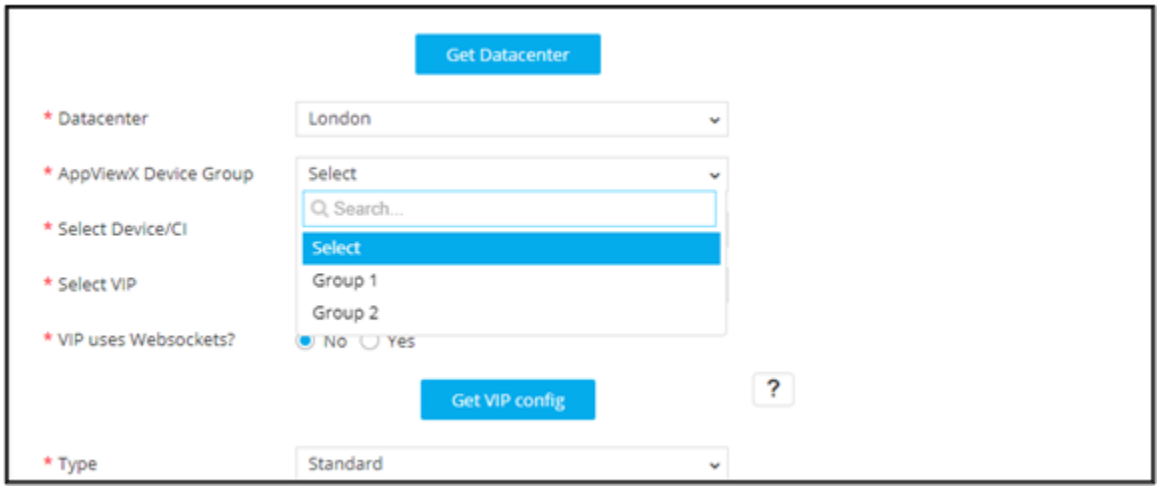
Field Name	Form Input/Response	Autotrigger script to execute
Get Datacenter	Get Datacenter - Button	getdataCenterlist
Datacenter	Response: Display list of datacenters in dropdown	getDeviceGrouplist
AppViewX Device Group	Response: Display List of Device Groups in dropdown	getDevicelist
Select Device/CI	ResponseL Display list of devices in dropdown	getVIPNames
Select VIP	Response: Display List of VIP Names for the device in dropdown	

2. To preview the form, from the menu on the top, click **(View)** icon.
3. In the form preview, click **Get Datacenter**.
   The script is auto triggered to populate the dropdown in the next (Datacenter) field.

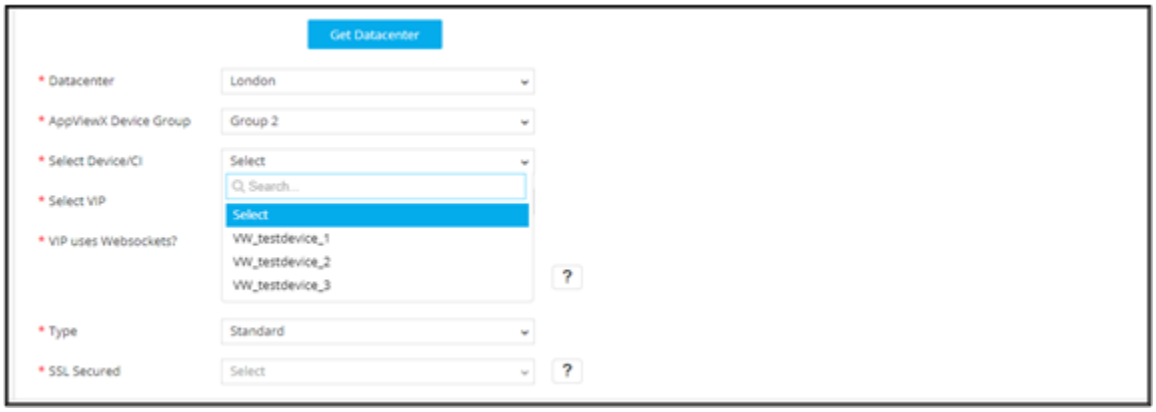
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. From the options displayed in the **Datacenter** dropdown list,select London from the dropdown list.
   The script is auto triggered to populate the next (device group) field.

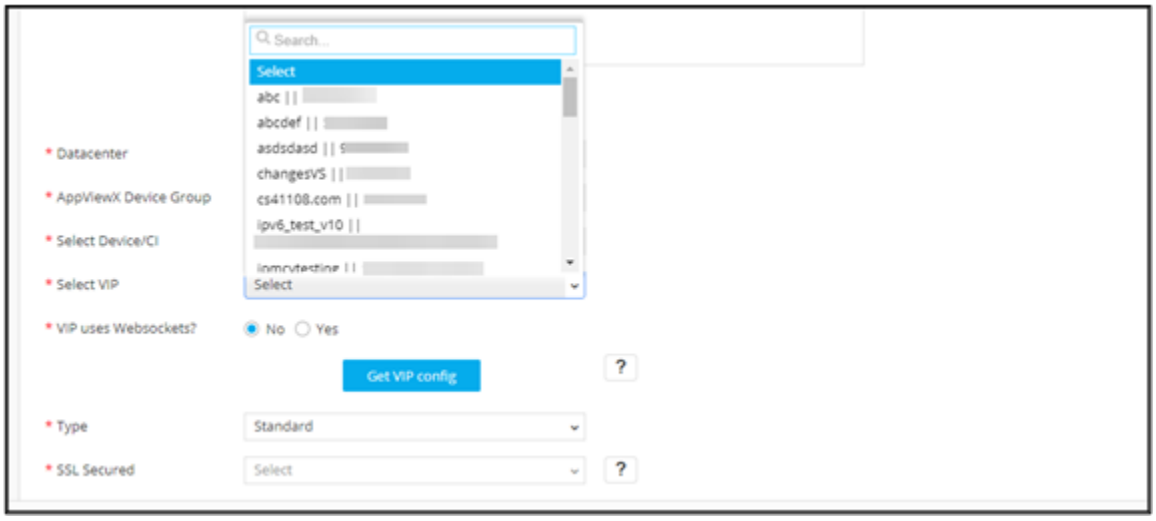


5. Select **Device group** as Group 2.
   The script is auto triggered to populate the next (select device) field.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. Select the **Device** to auto trigger the script and populate the next (select VIP) field.



Note: Hooks can be executed by an auto-trigger even if the fields are in different form groups.

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.

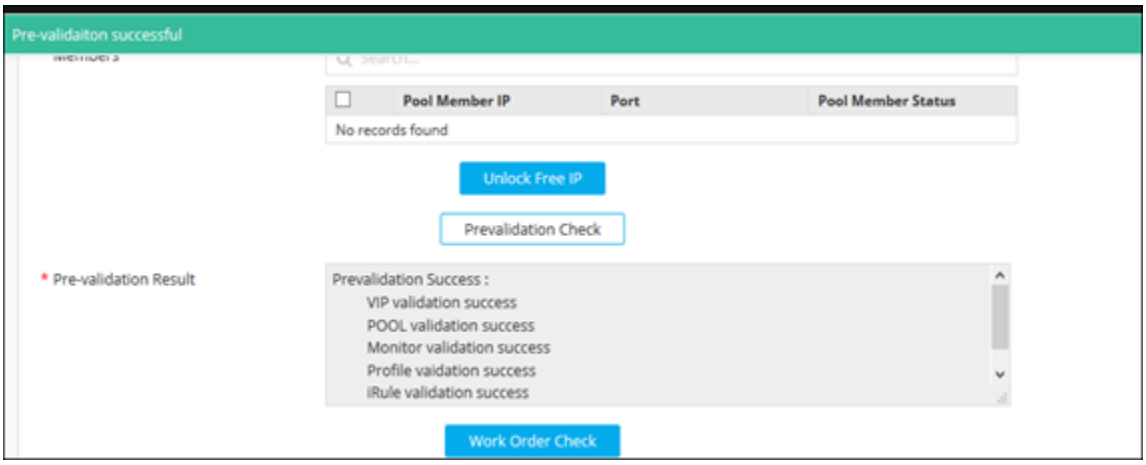
For example, As part of a Create DNS workflow, a user gets a Free IP, reserves it, and then immediately wants to cancel the request transaction. In this case, the reservation of IP and any other transactions will need to be ‘cleaned up’. The Cancel script Logic releases any temporarily locked free IP and releases any reservation on the IPAM in the event of a cancel or discard action on the form.

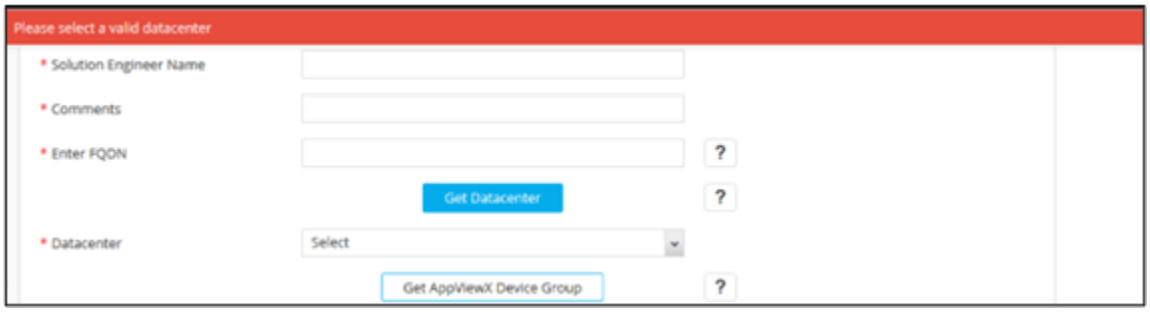
Note: Default associate scripts for ‘Cancel’ and ‘Discard’ cannot be deleted.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Define Custom 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.





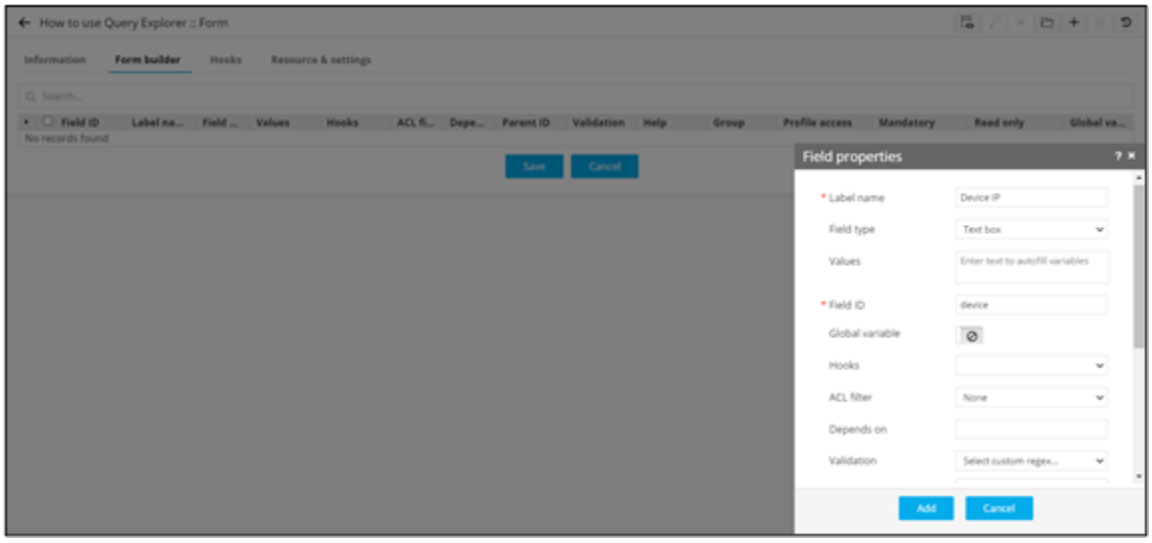
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Defining Query Explorer to Build Forms

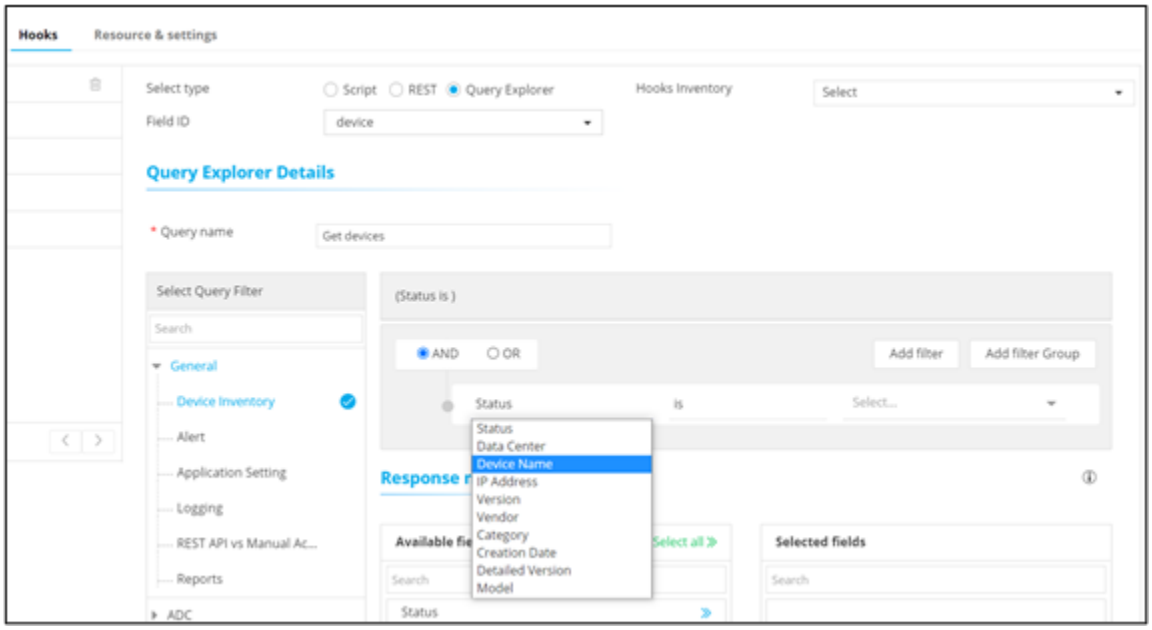
Query explorer allows you to design custom queries, business logic, and build self-service forms quickly.

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.

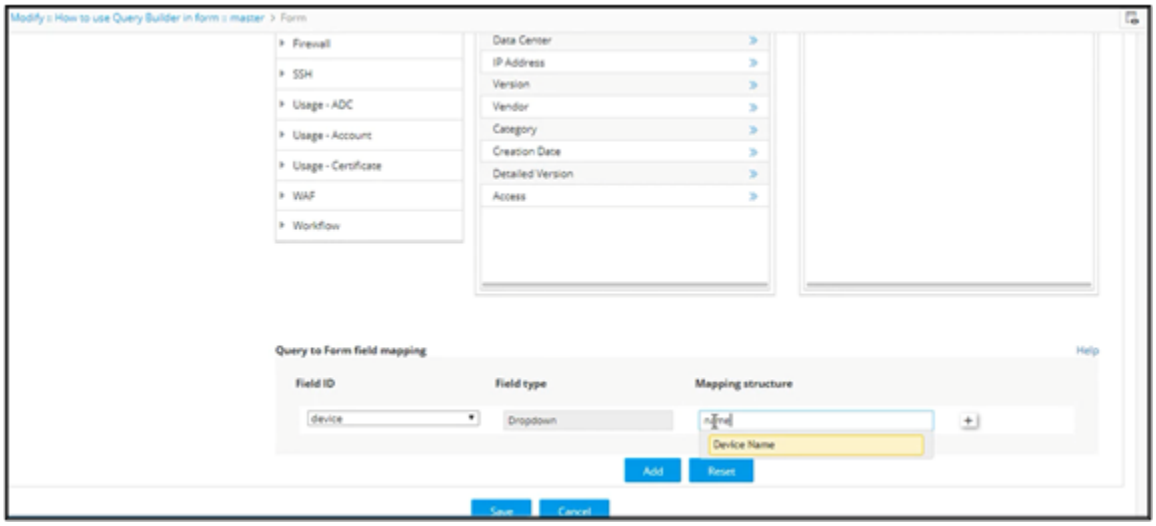


4. Under **Hooks**, define the required Hook using Query explorer.
5. Select the required response from the query to be mapped to the form field.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. Map the query response to the specific form field.

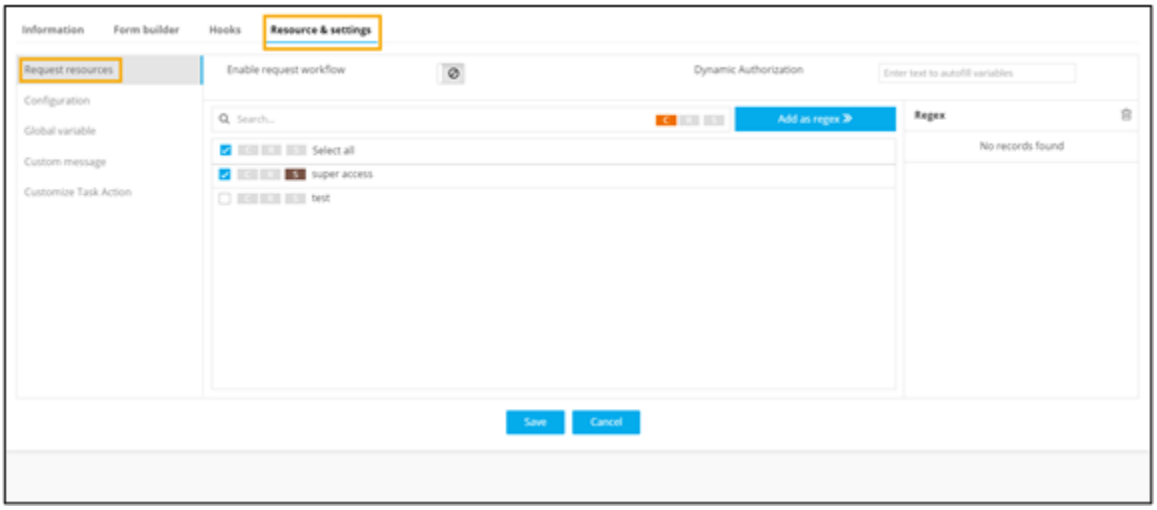


7. To validate the execution, click **(View)** icon.

Defining Resource & Settings

This section supports the following:

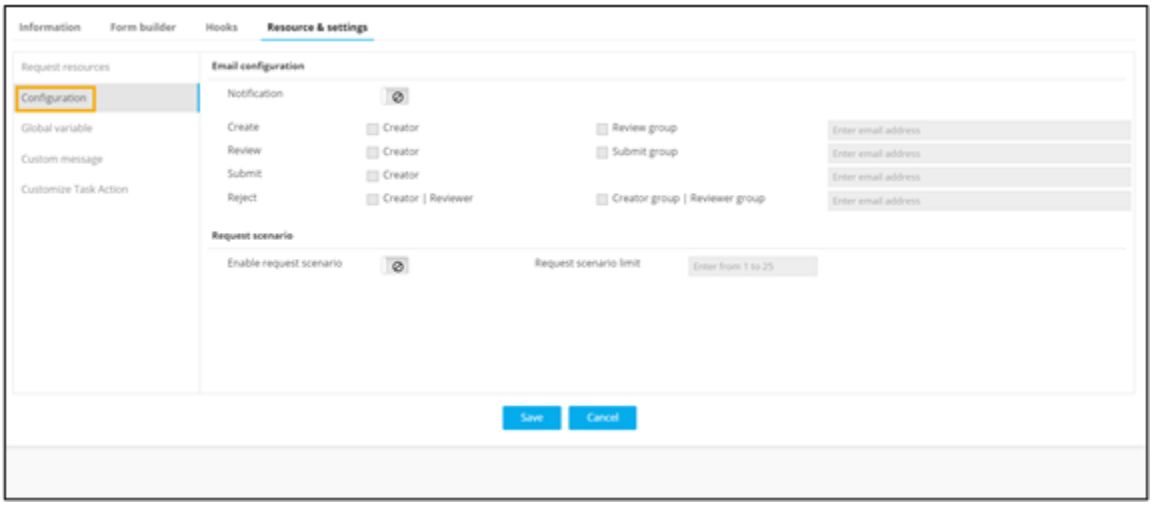
- **Request resources**: Provision to assign role based access to the form task.



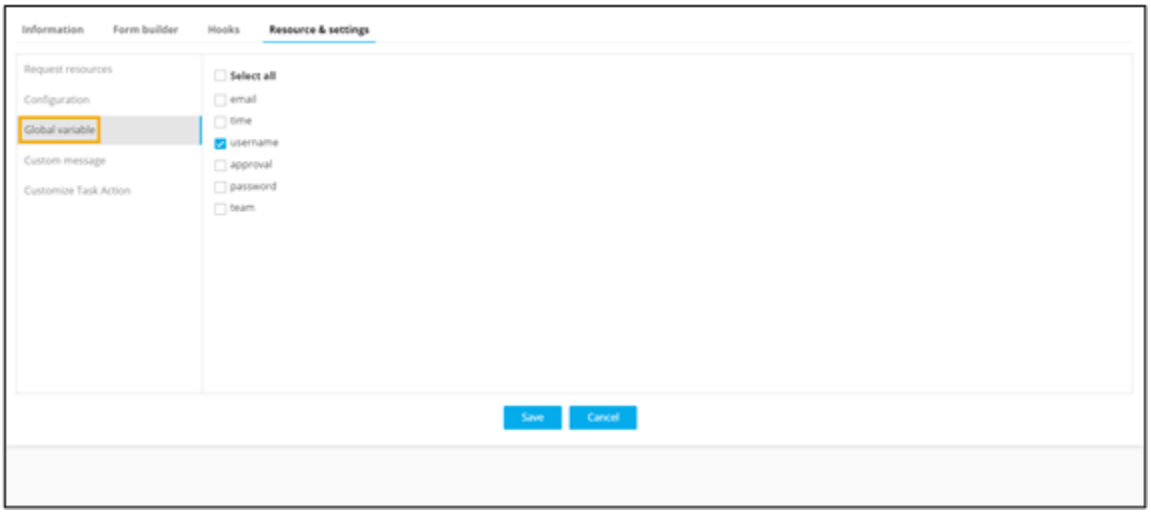
When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.

- **Configuration**: Provision to enable multiple approvals of the form task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

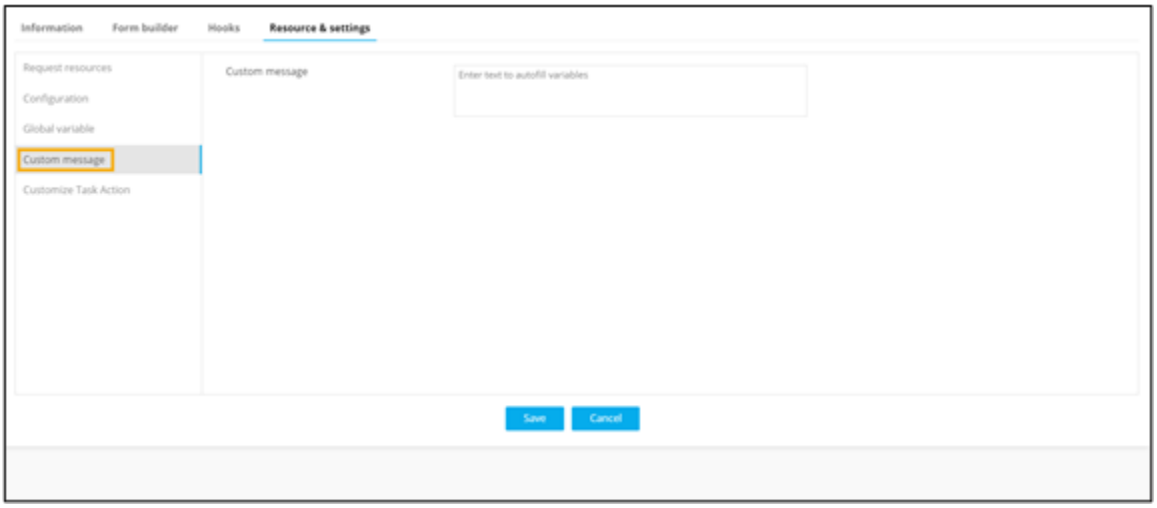


- **Global variable**: Provision to declare one or more form field values as global variables.

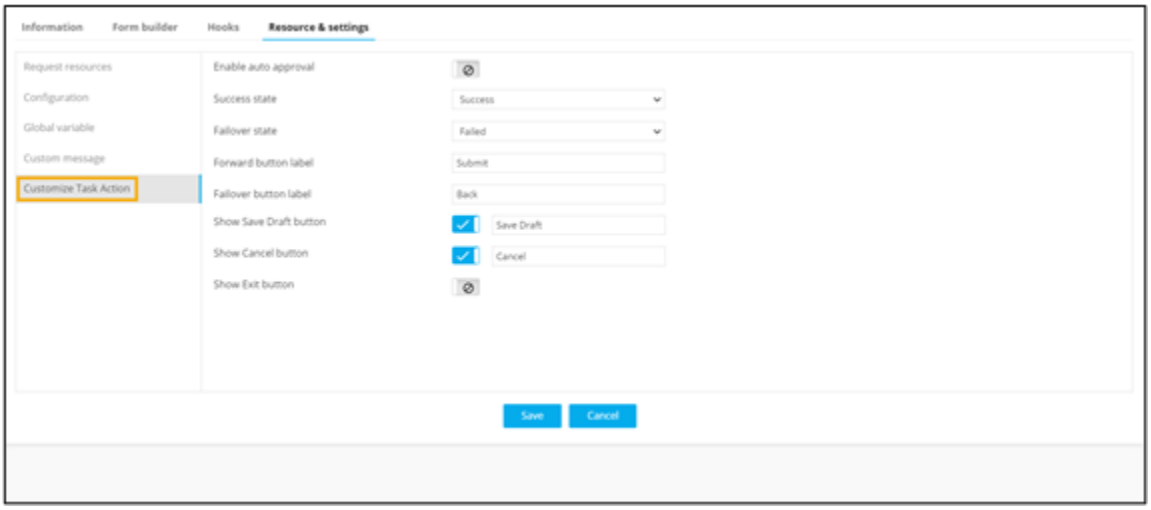


- **Custom Message**: Provision to define context based messages for notifications between workflow stages. For example, User access denied.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- **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’.



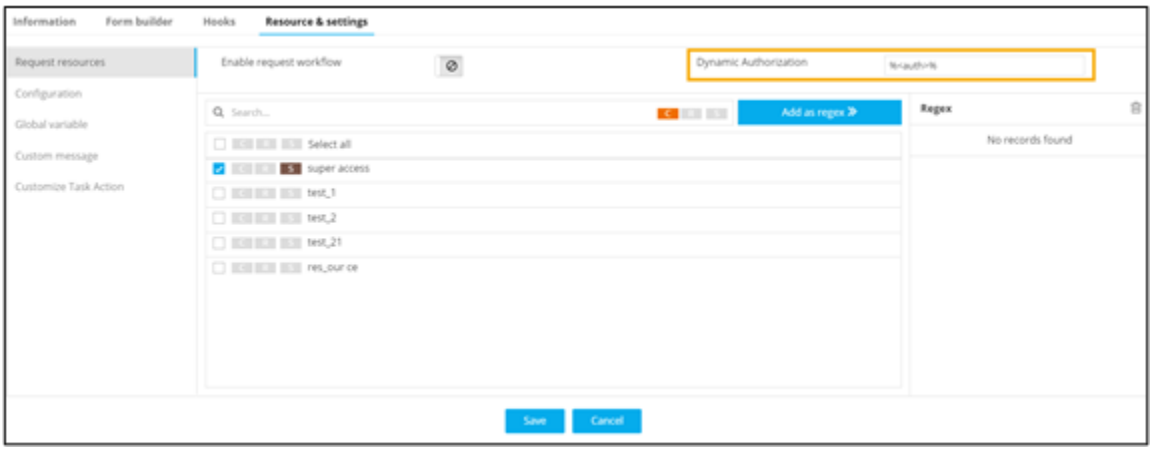
- **Dynamic Authorization**: Provision to enable authorization dynamically for users or user groups by using the global variable `%<auth>%`.

  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"},
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

{"type":"usergroup","value":["admin usergroup"],"access":"RW"}]



Note: Dynamic Authorization is not limited to the Form task. It can be enabled for all User Interface tasks.

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.

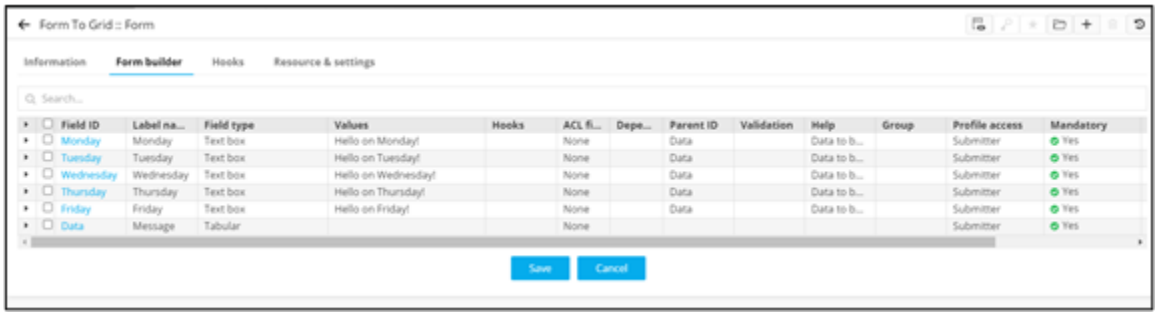
- Provision to define a custom grid component with number of columns.
- Provision to define custom column/label names.
- Provision to dynamically pass data using variables from any stage of the workflow into the grid task.
- Provision to assign user role RBAC access.
- Provision to define custom button/label 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.

- Downloading the Grid

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

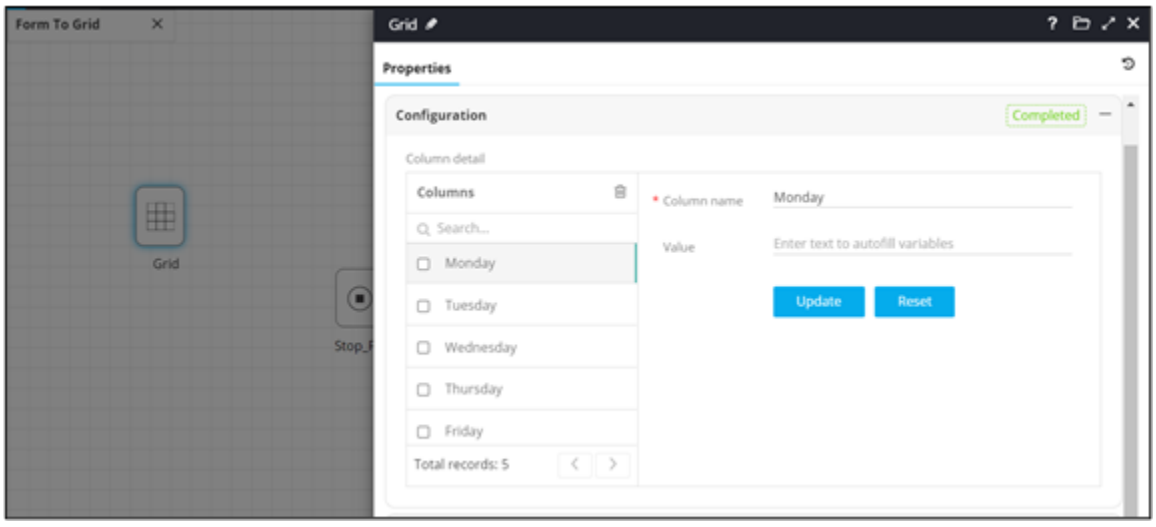
Downloading the Grid

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.

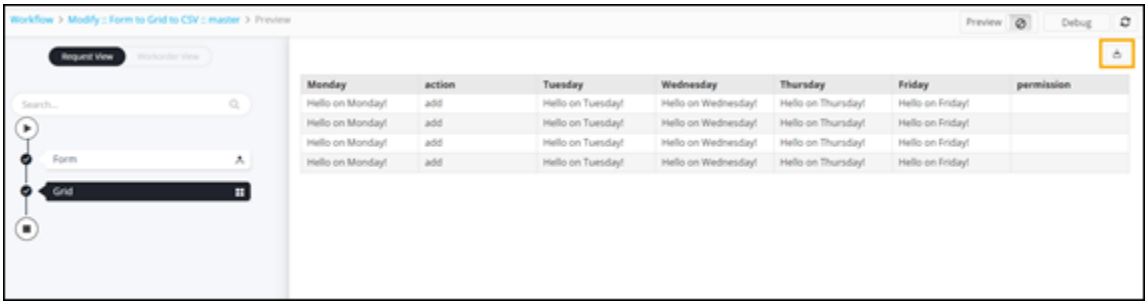
4. From the **User Interface** section, drag and drop the **Grid** task.
5. In the **Grid** task window, under **Properties**, in the **Configuration** section, configure the **Columns** for the grid.



6. In the **Grid** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
7. Connect the workflow tasks and from the menu on the top, click **Preview**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

8. To download the Grid, click **(Download)** icon in the right upper corner of the screen.



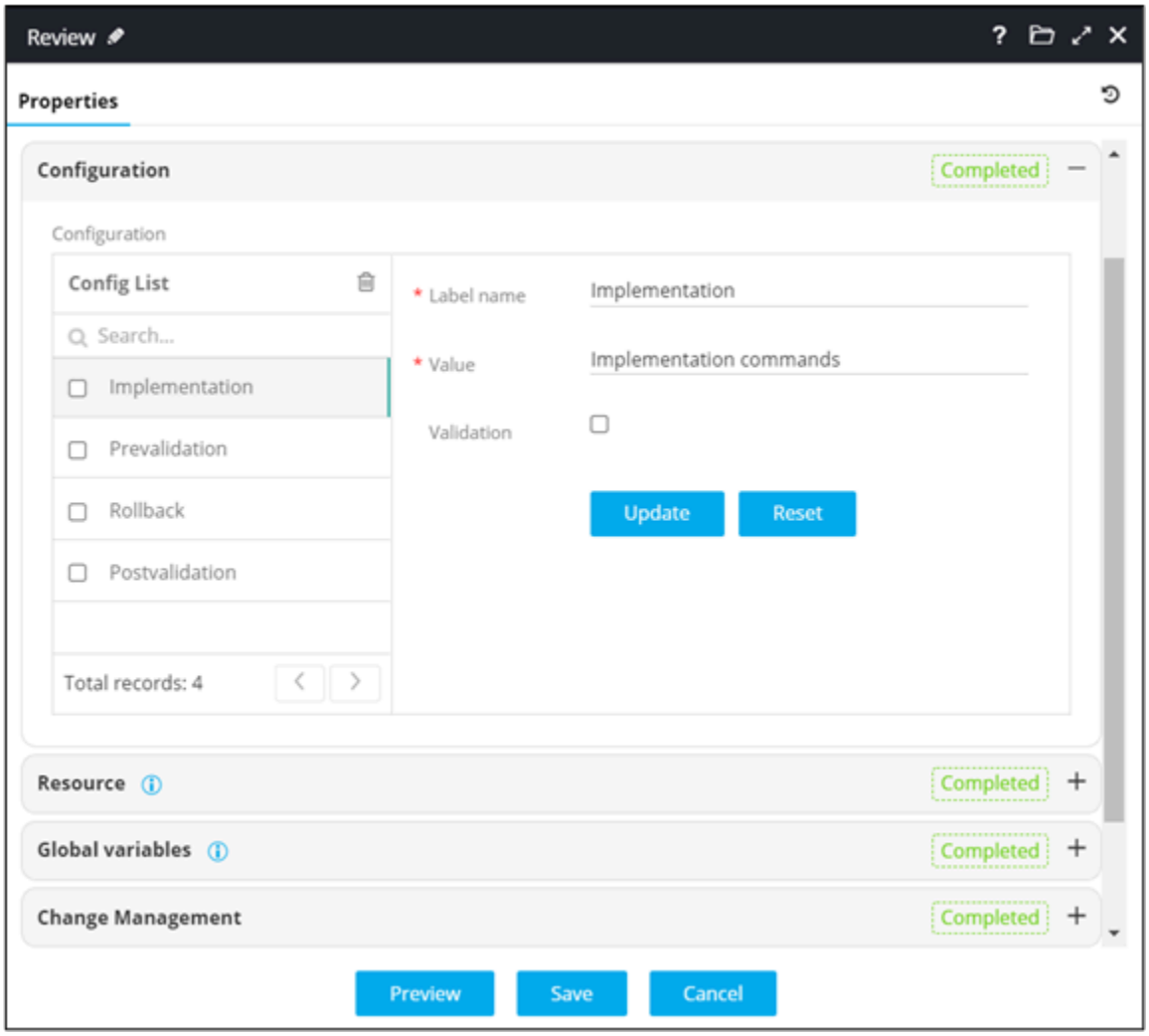
Review

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’
- 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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



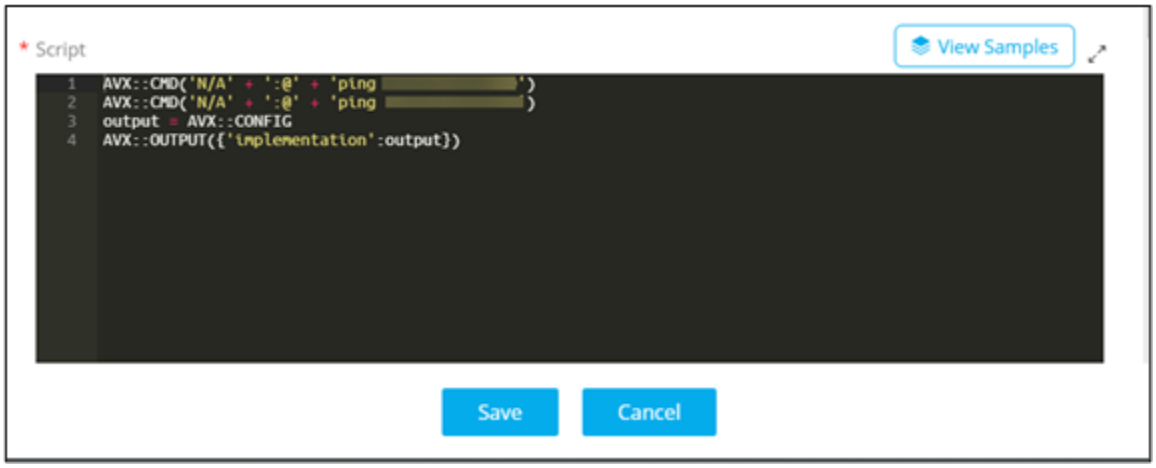
- Defining a Review Task using Variables
- Customizing Configuration Blocks in Review Task

Defining a Review Task using Variables

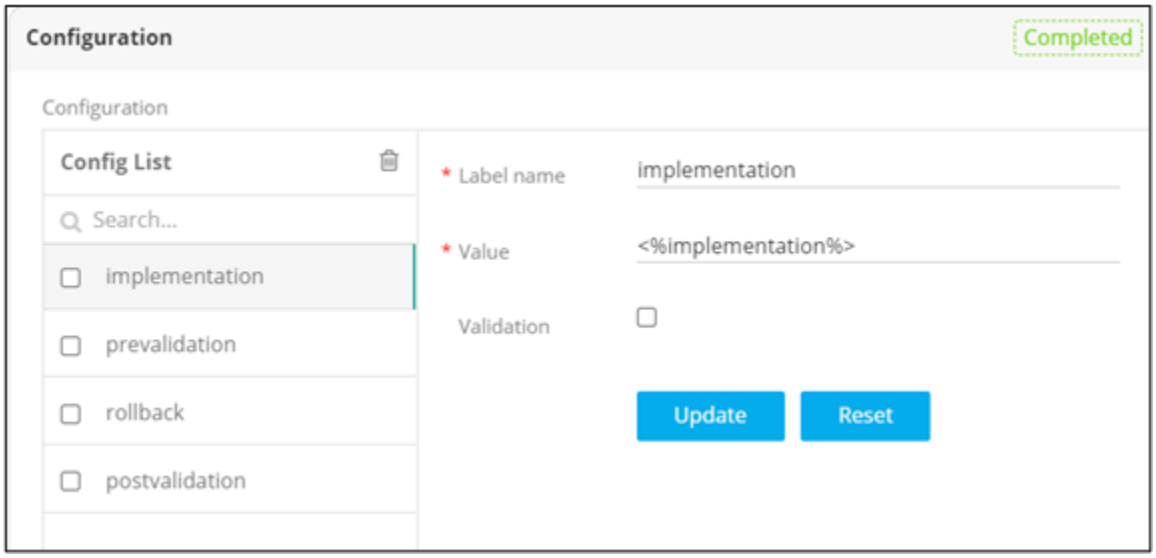
A Review component can be defined using either static or dynamic values.

1. Design a workflow.
2. In the design space, 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 and store the configuration in a variable (AVX::CONFIG).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



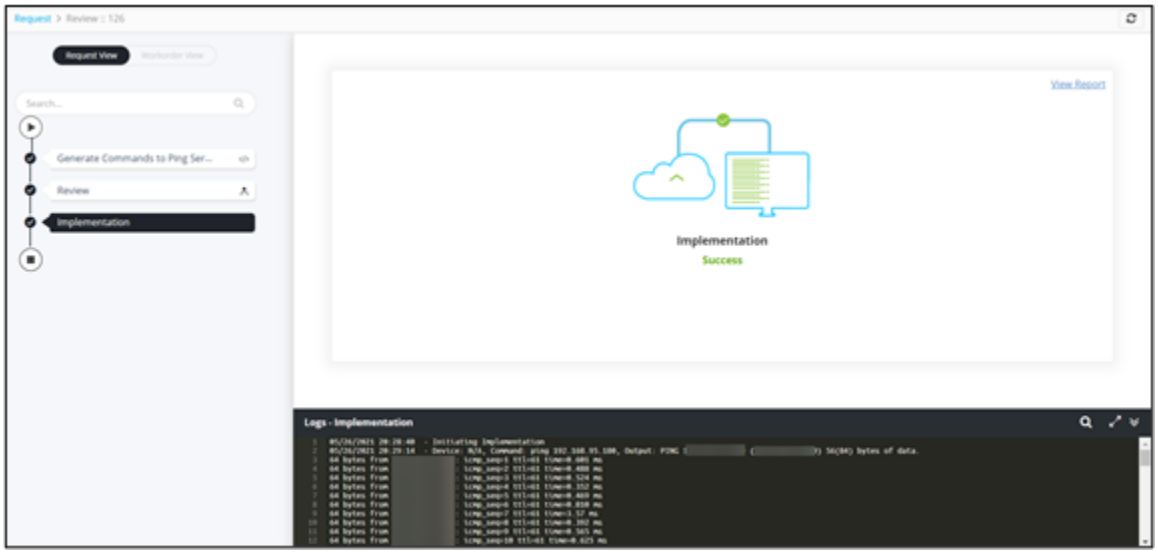
4. From the **User Interface** section, drag and drop the **Review** task.
5. In the **Review** task window, under **Properties**, in the **Configuration** section, define the configuration blocks - Prevalidation, Implementation, Rollback, Postvalidation.
6. Refer the variables from the Script task into the appropriate configuration blocks.



7. From the **General** section, drag and drop the **Implementation** task.
8. In the **Review** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
9. Connect and enable the workflow.
10. Trigger the workflow from the Request :: View/Run page.

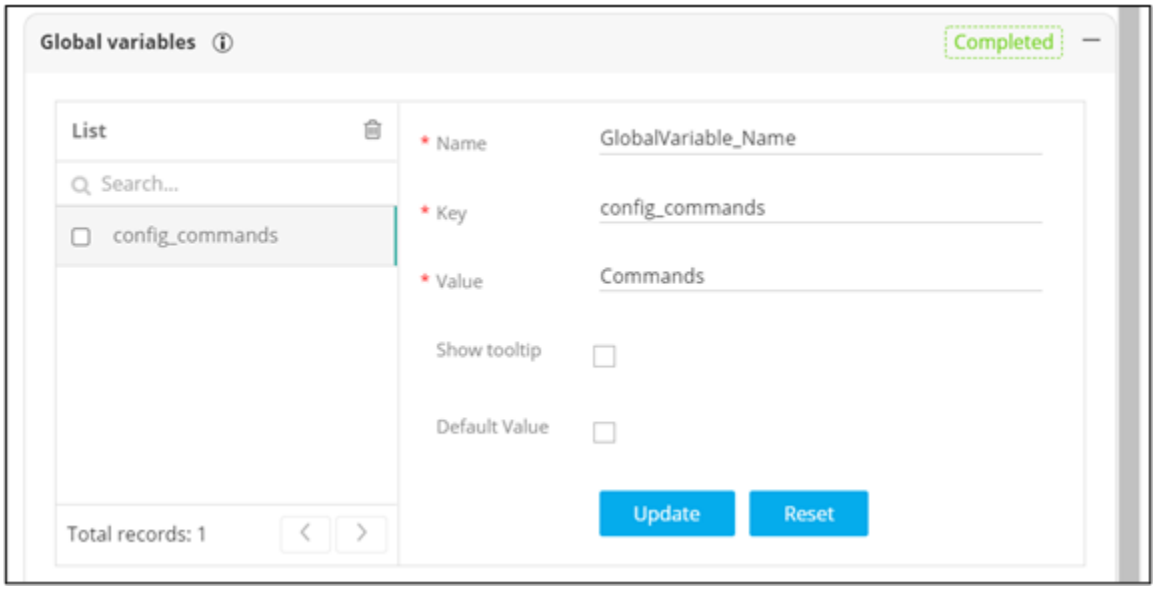
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Workflow is executed successfully with the Review task.



Customizing Configuration Blocks in Review Task

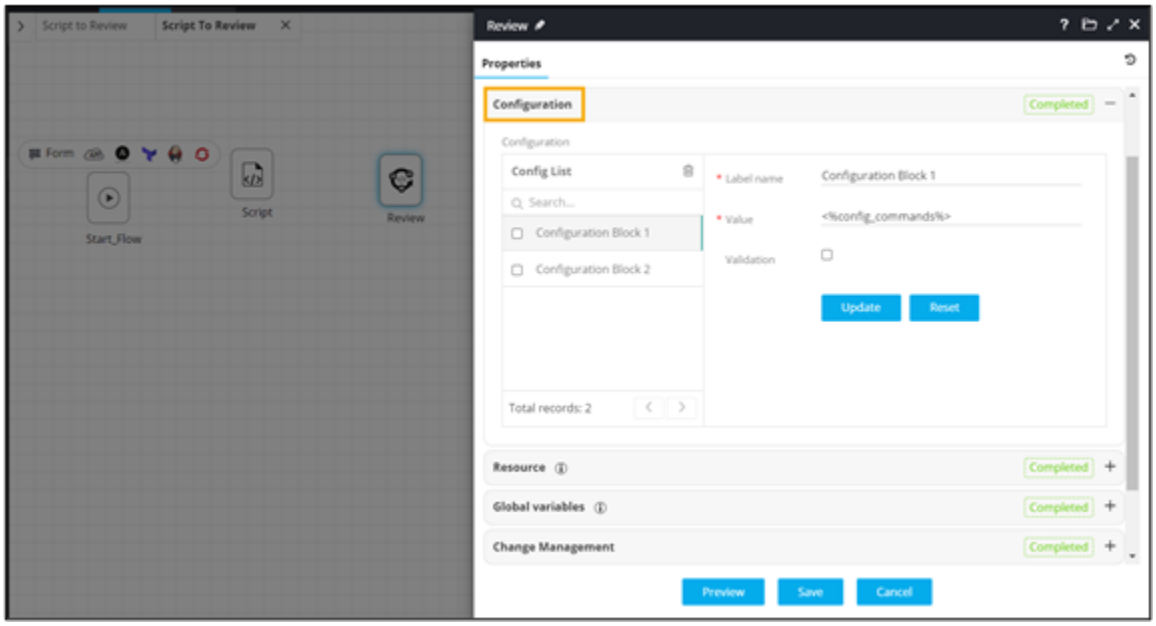
1. Design a workflow.
2. From the **General** section, drag and drop a **Script** task.
3. In the **Script** task window, under **Global variables**, declare the data to be passed as a global variable to be referenced in the Review task.



4. From the **User Interface** section, drag and drop the **Review** task.
5. In the **Review** task window, under **Configuration** , define custom configuration blocks.

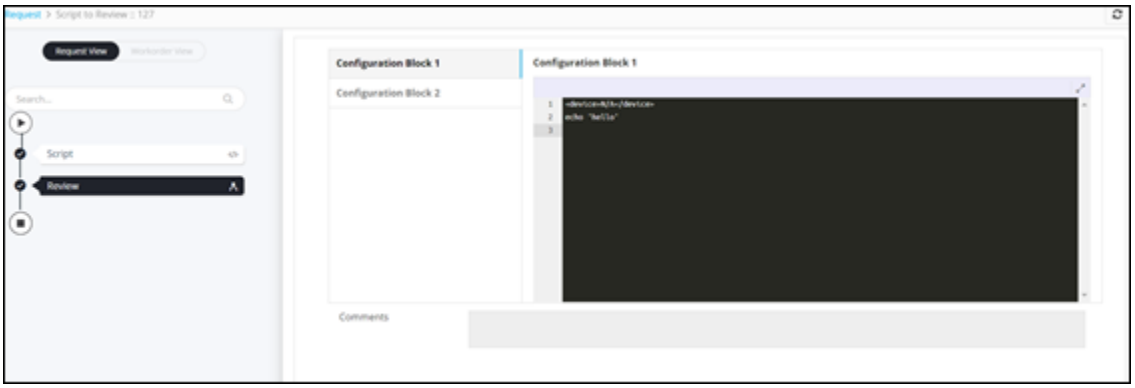
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. Pass variable(s) from the previous task in the Value field against the appropriate configuration block.



7. Connect and enable the workflow.
8. Trigger the workflow.

   Workflow is executed successfully with the customized configuration blocks.



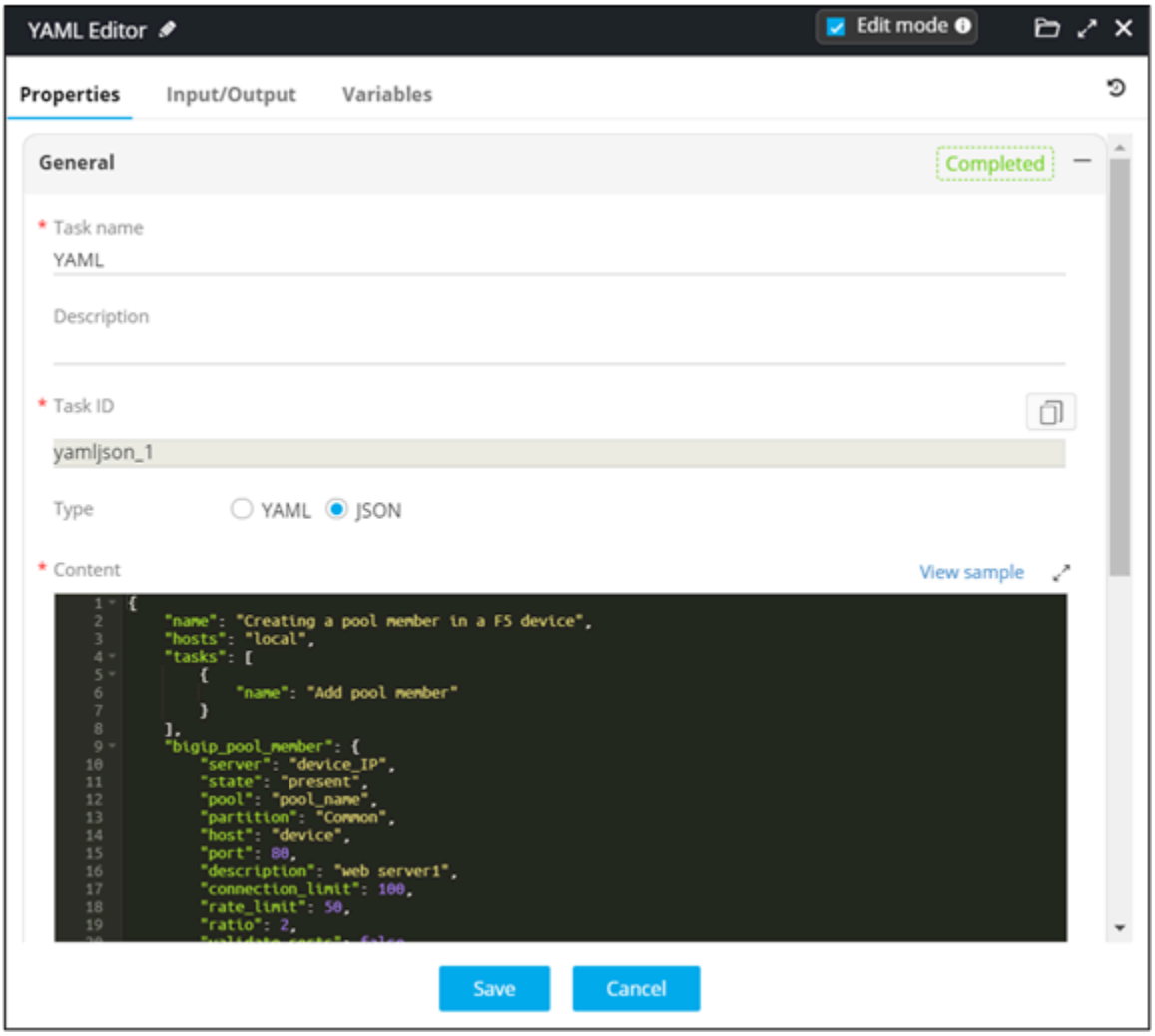
YAML

This task allows you to input the workflow data in the form of YAML or JSON content.

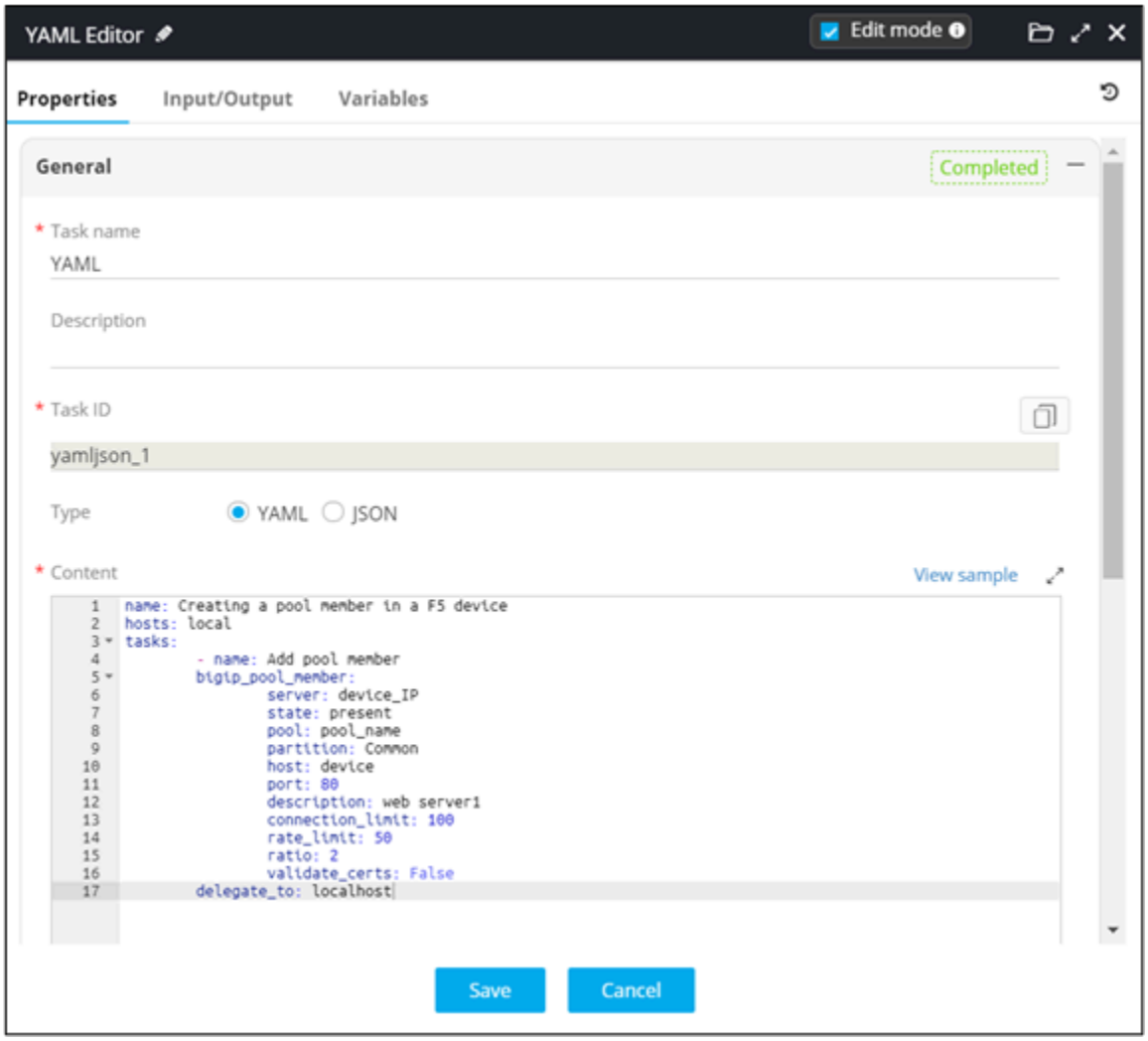
- Provision to input data in either YAML or JSON format
- Provision to refer YAML or JSON samples
- Provision to use the YAML task as either a user interface task or as a service task
- Provision to edit the contents of the task in the request page

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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 © 2025 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:

```
--

-

 name: Creating a virtual server in a F5 device
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
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
```

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:

```
--
-
 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: $$<%device%>.password$$
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    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.

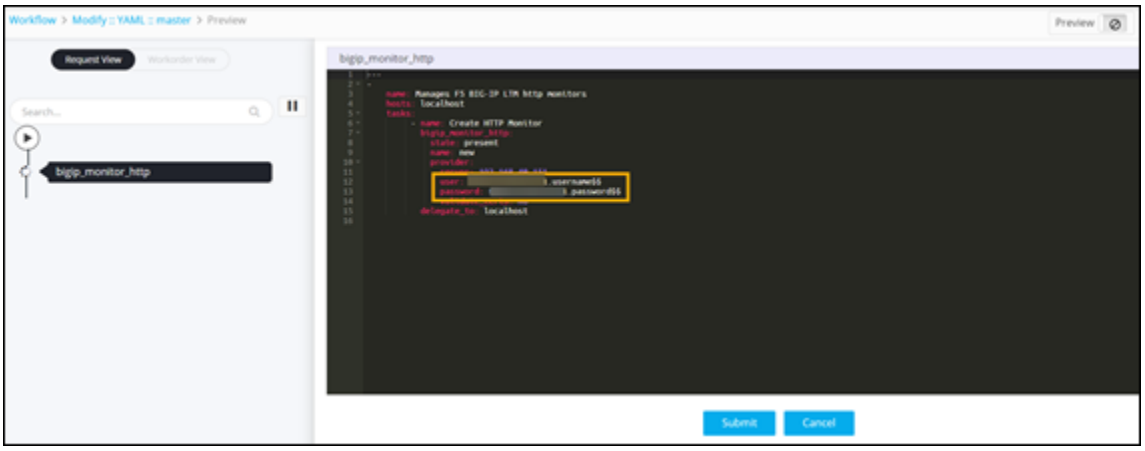
To get the device credentials dynamically:

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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. In the **YAML** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
5. Connect the workflow tasks and enable the workflow.
6. Trigger the workflow.

   Device credentials are dynamically pulled from the Inventory.



Holistic View

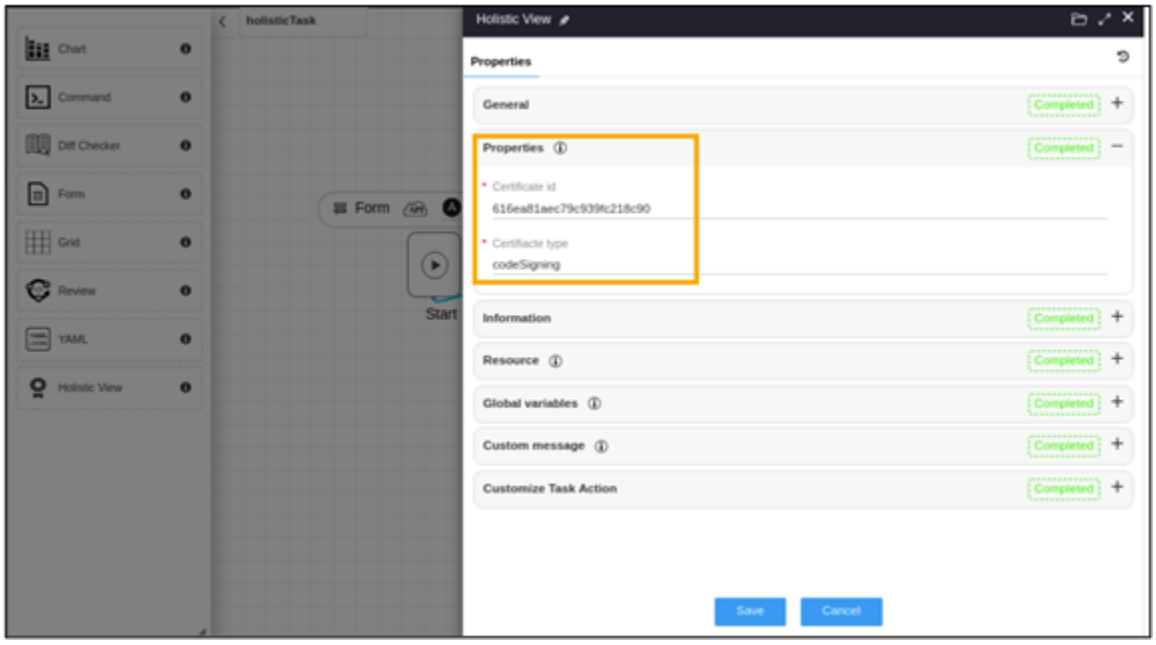
This task allows you to view the certificates in a holistic view in the workflow Studio. You have the option to download the certificate and the certificate key from the Holistic View stage of the workflow.

To add Holistic View task to your workflow:

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

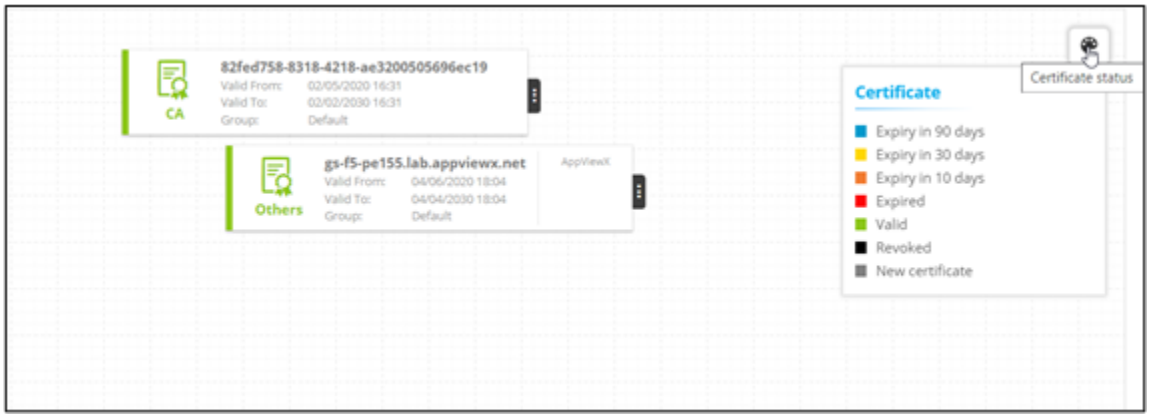
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. In the **Holistic view** task window, under **Properties**, define the **Certificate id** and **Certificate type** either as static values or a global variable.



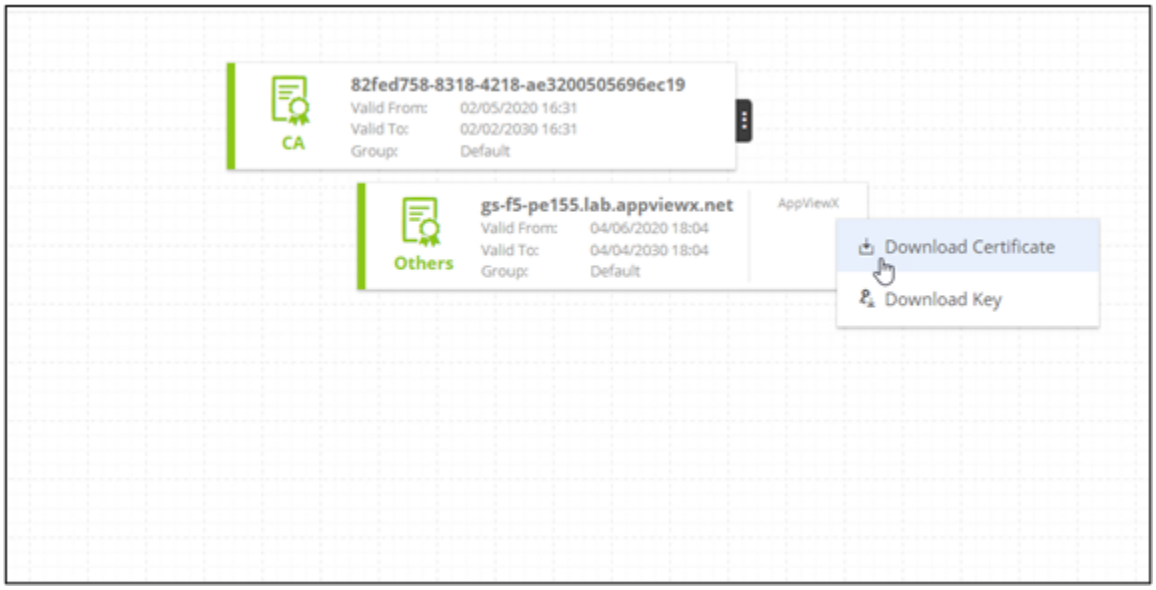
4. In the **Holistic view** task window, under **Resource**, define the permissions in the chart as follows:
   - **V** (View) - Users can only view the interface. There is no Submit or Cancel buttons provided.
   - **R** (Review) - Users can only view/read, Submit and Cancel without editing the task.
   - **RW** (Read Write) - Users can read/write, Edit, Submit, and Cancel the task.
5. Connect the workflow tasks and from the menu on the top, click **Preview**.

   Holistic view is executed. Hover your mouse over **(Certificate status** icon to view the certificate status.



6. To download the certificate and certificate key, click .

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



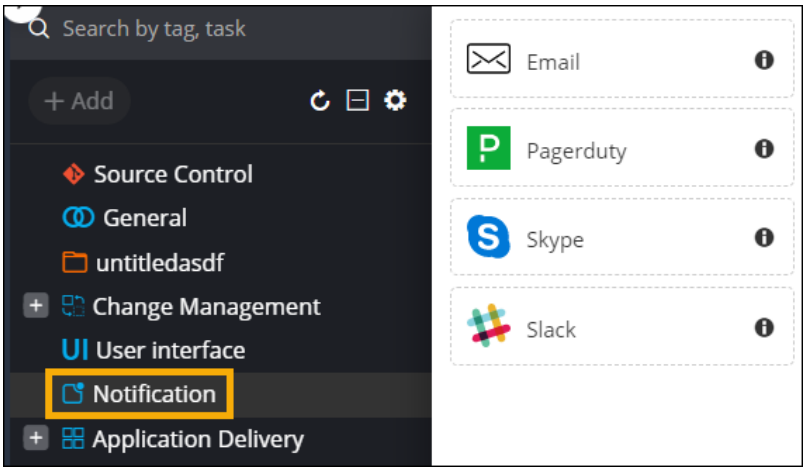
7. Click **Submit**.

Note: You can add multiple Holistic view tasks to the workflow.

Note: For more information on Holistic View, refer to the Cert+ User Guide.

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.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Email
- Pagerduty
- Slack
- Skype

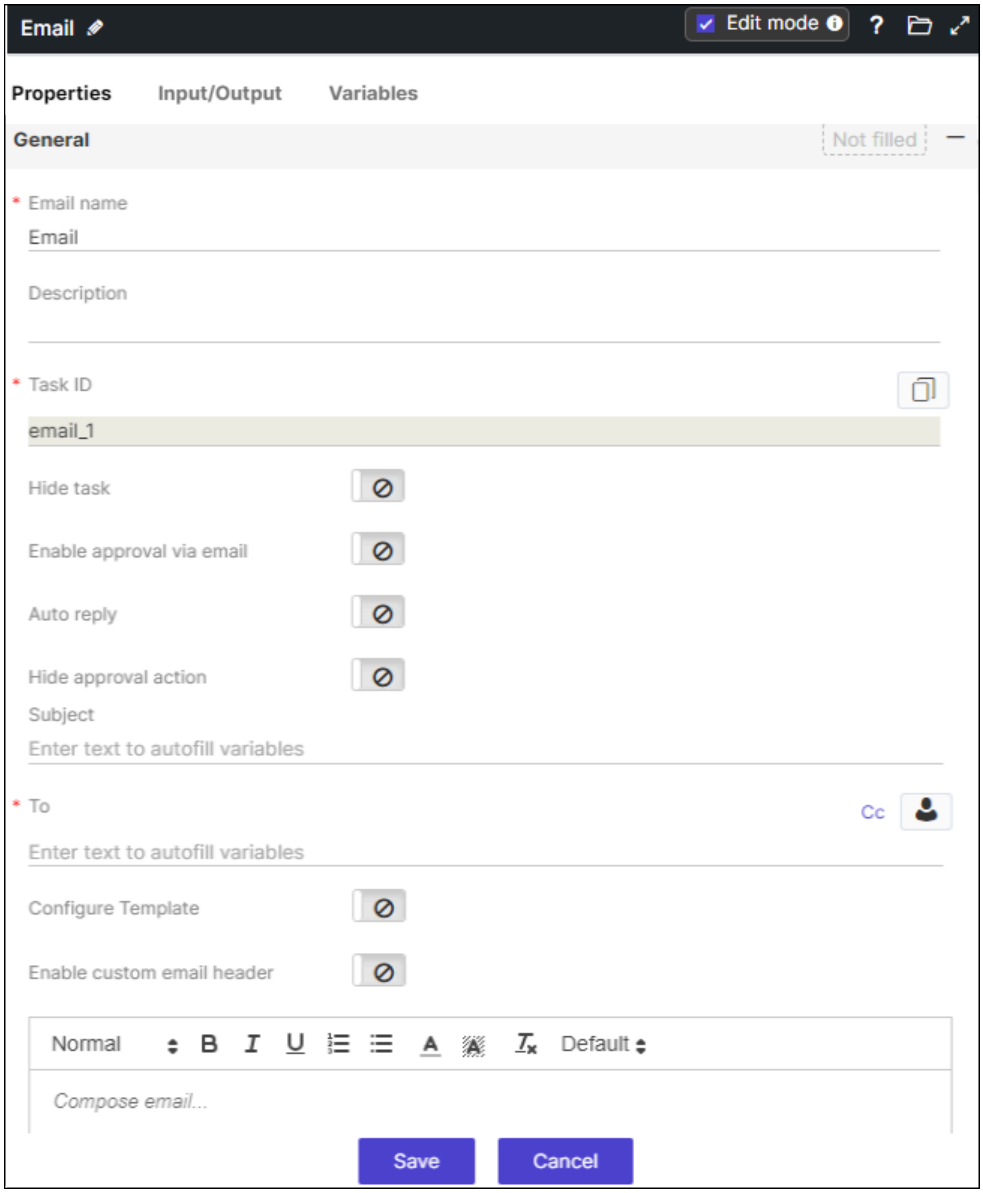
Email

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 © 2025 AppViewX, Inc. All Rights Reserved.

- Provision to use customized email template within your workflow to send emails.



- Configuring Dynamic Email Attachments
- Configuring Email Approvals
- Receiving Approvals via Email
- Triggering an Email Notification with Approvals
- Configuring Email Template
- Customizing Email Notification

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

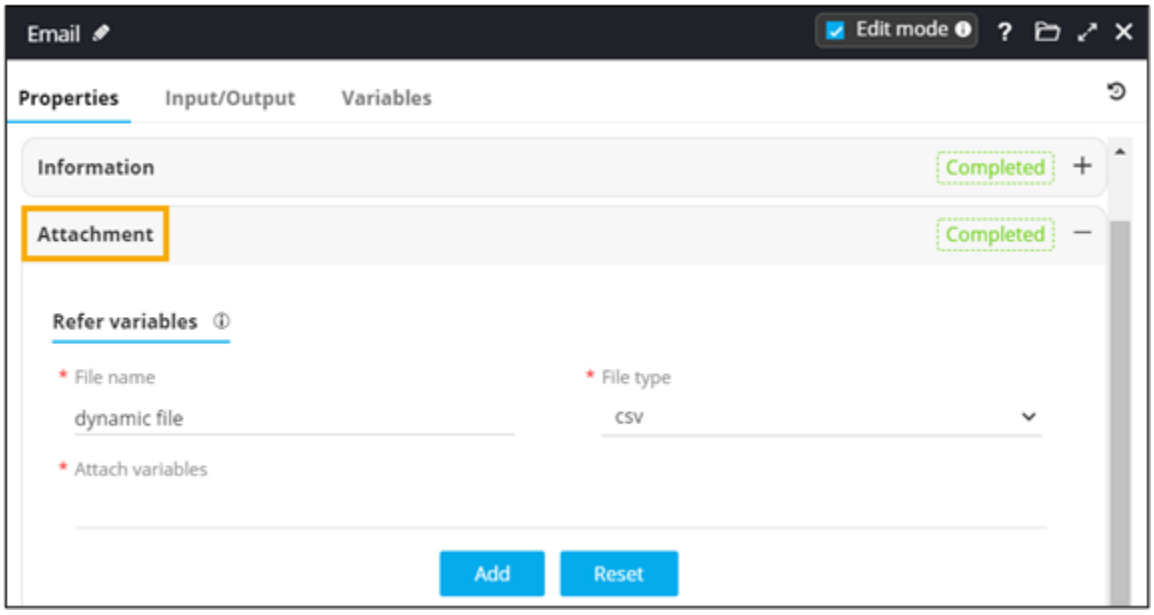
Configuring Dynamic Email Attachments

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.

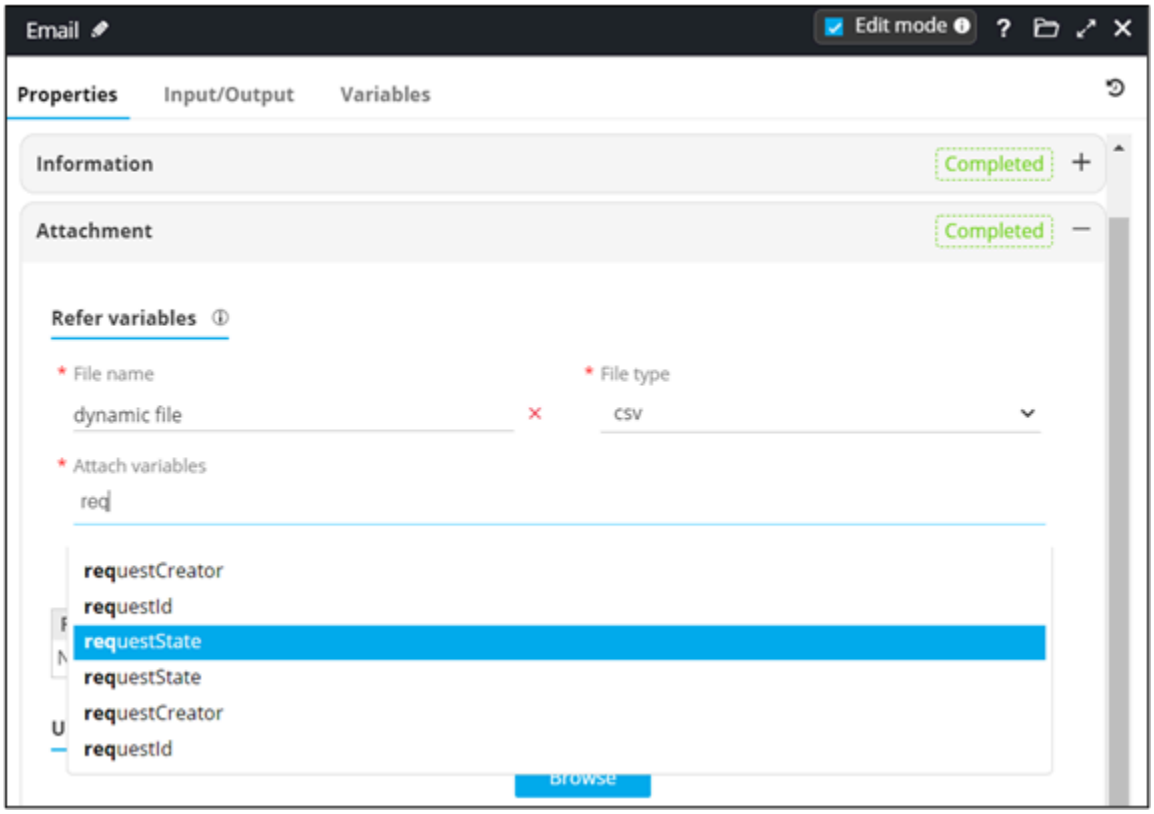
To send an attachment:

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 © 2025 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.

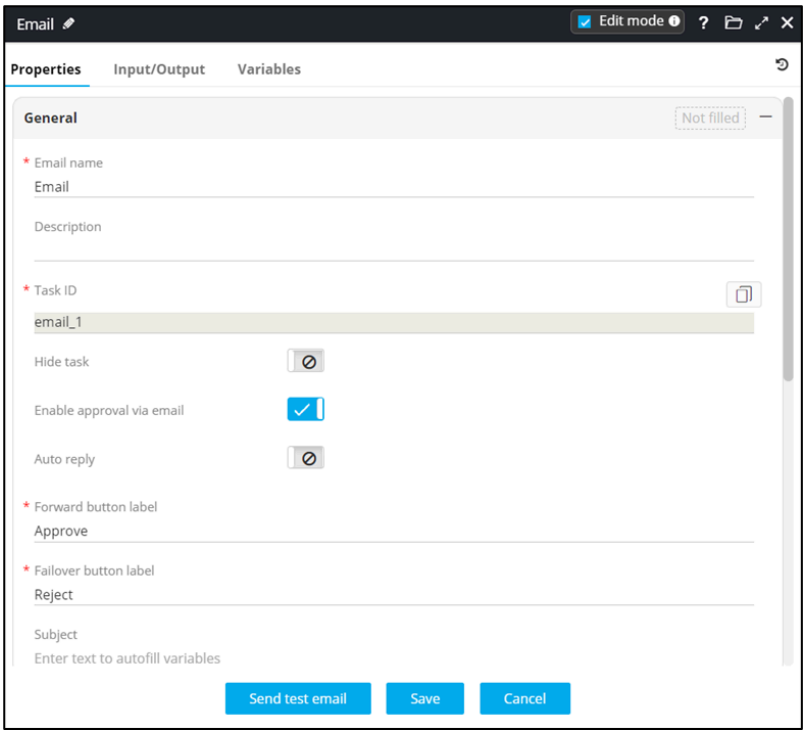
- Configure approvals via email and trigger the workflow based on the email event or action.
- Define manual or automatic email approvals.
- Emails include static and dynamic file attachments (pdf, jpeg, doc, xls).

To enable email approval:

1. Design a workflow.
2. From the Notificationsection, drag and drop the **Email** task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. In the **Email** task window, under **Properties**, in the **General** section, turn on the **Enable approval via email** toggle.



On enabling the toggle, the following fields are displayed:

- **Forward button label**
- **Failover button label**.

> **Note:** You can also use **Configure Template** within your workflow to send email approvals using your customized email template.

4. Enter values for these two fields.
5. Enter custom email content.
6. To assign users and user group, in the **General** section, go to the **To** field and click on the person/ recipients icon.
7. In the **Choose Recipients** icon select **User Type** as *User/Role/Group*. Based on the selection further click the check boxes for users/ roles/ user groups to be added.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

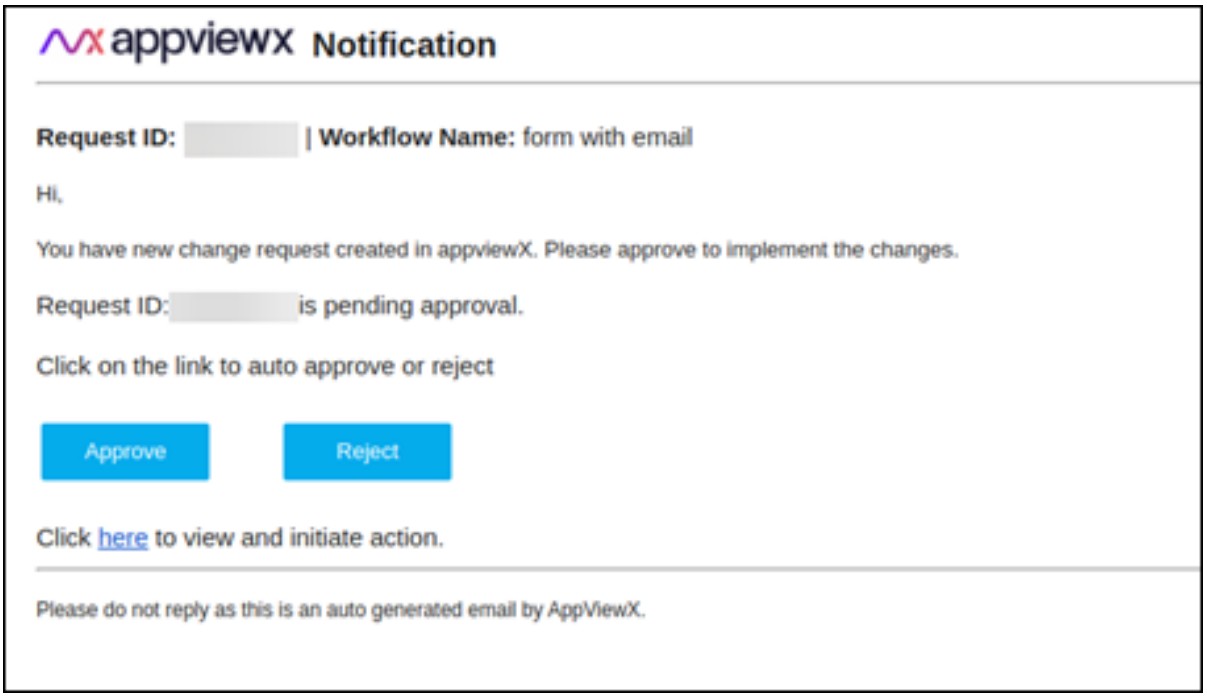
Note: In case of User Groups, emails are sent based solely on the user's direct association with the selected user group. If User1 is associated with UserGroup1 and User2 is associated with UserGroup2. Consequently, if UserGroup1 is linked to Role1 and Resource1 and UserGroup2 is linked to Role1 and Resource2. User1 receives the email as they are directly associated with UserGroup1, and User2 does not receive the email since they are not part of UserGroup1.

8. Click **Save**.
   You can use this task within your workflow to send email approvals.

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.

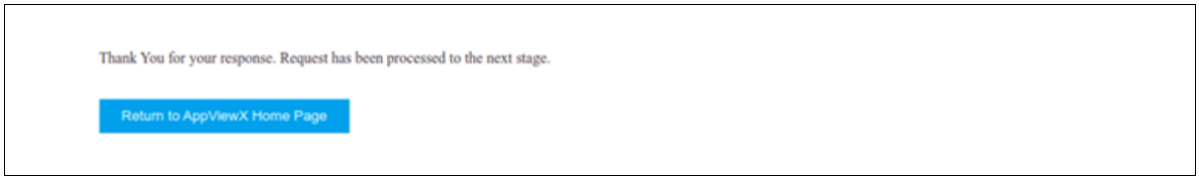
1. Click on the link embedded in the approval email (Approve/Reject) to navigate to AppViewX.



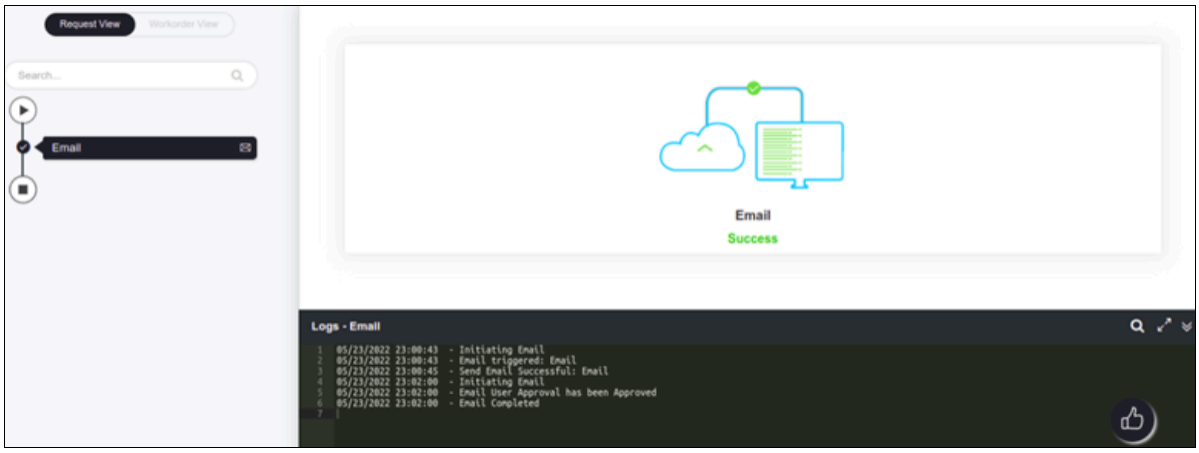
2. Click **Approve** / **Reject**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

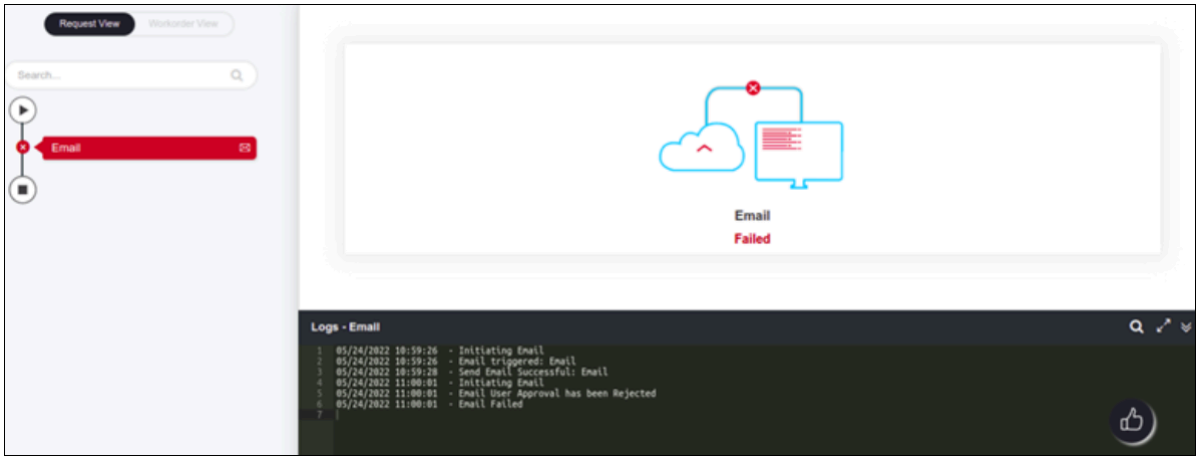
- The following message (HTML text) will be shown in a new tab.



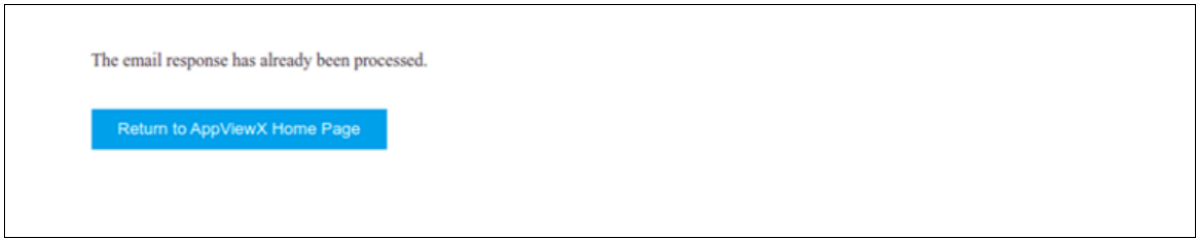
- If you click **Approve**, the task is displayed as **Success**.



- If you click **Reject**, the task is displayed as **failed**.

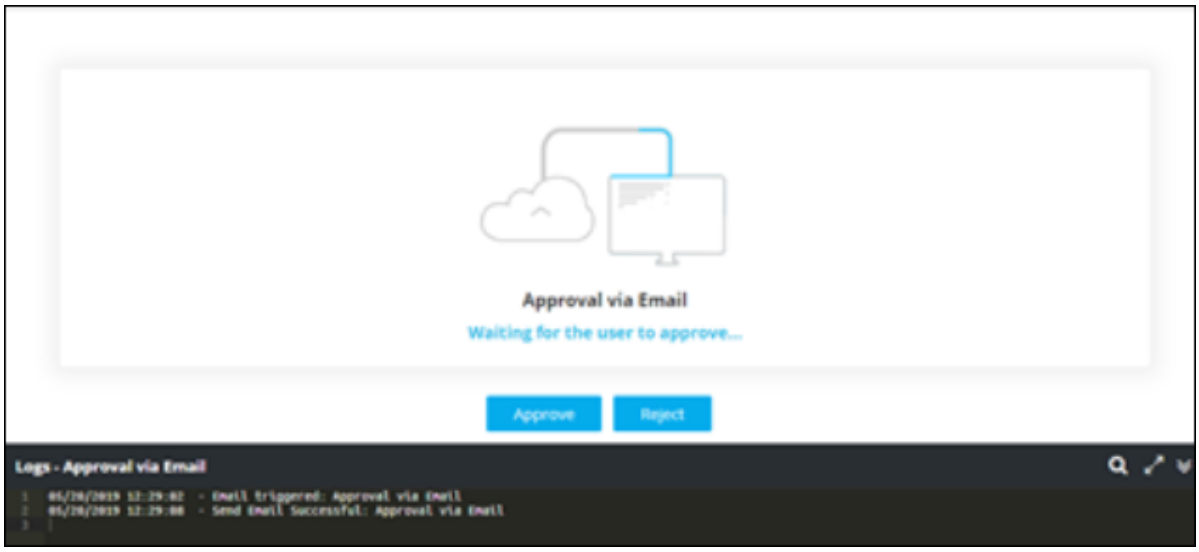


- In the event of clicking the email approval again, the following message indicating that the email response has been processed will be shown in a new tab.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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.



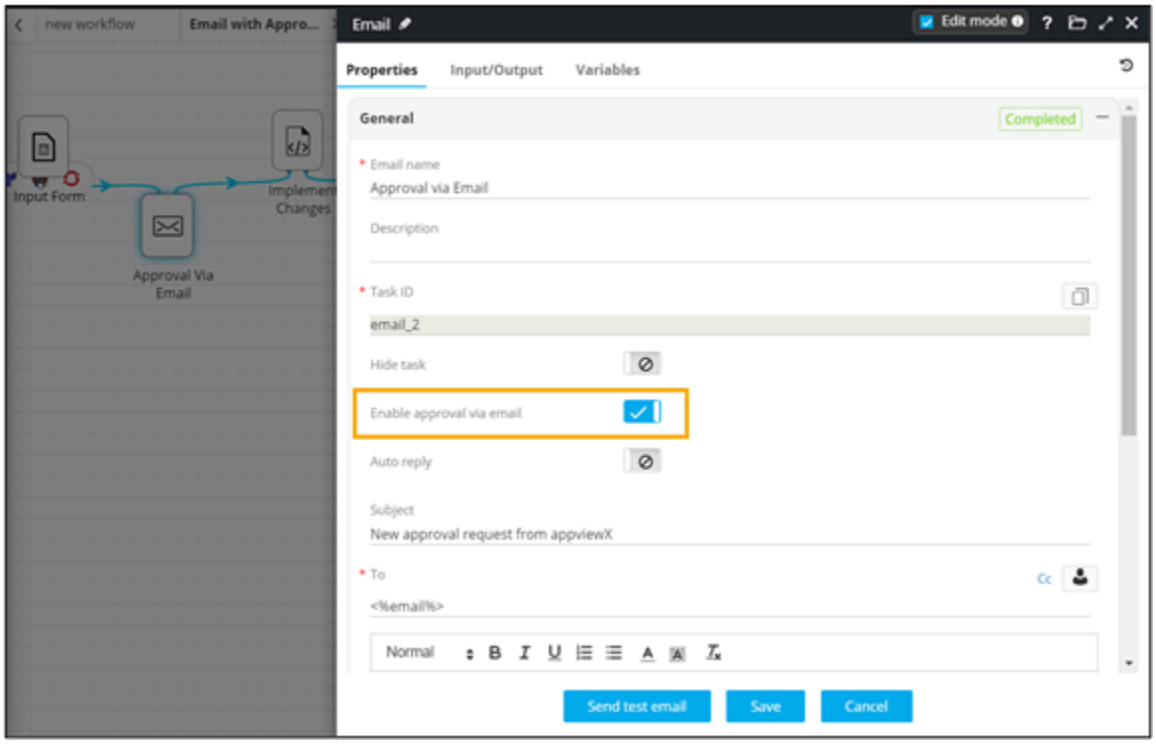
Note: The SMTP settings must be configured prior to triggering any emails. For more information, refer to the Platform User Guide.

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.
2. From the FAQs section drag and drop the > **Email with approval** workflow.
3. Double-click on the workflow to see the tasks within the workflow.
4. Click on the **Approval Via Email** task in the workflow.
5. In the **Email** task window, under **Properties**, in the **General** section, turn on the **Enable approval via email** toggle.

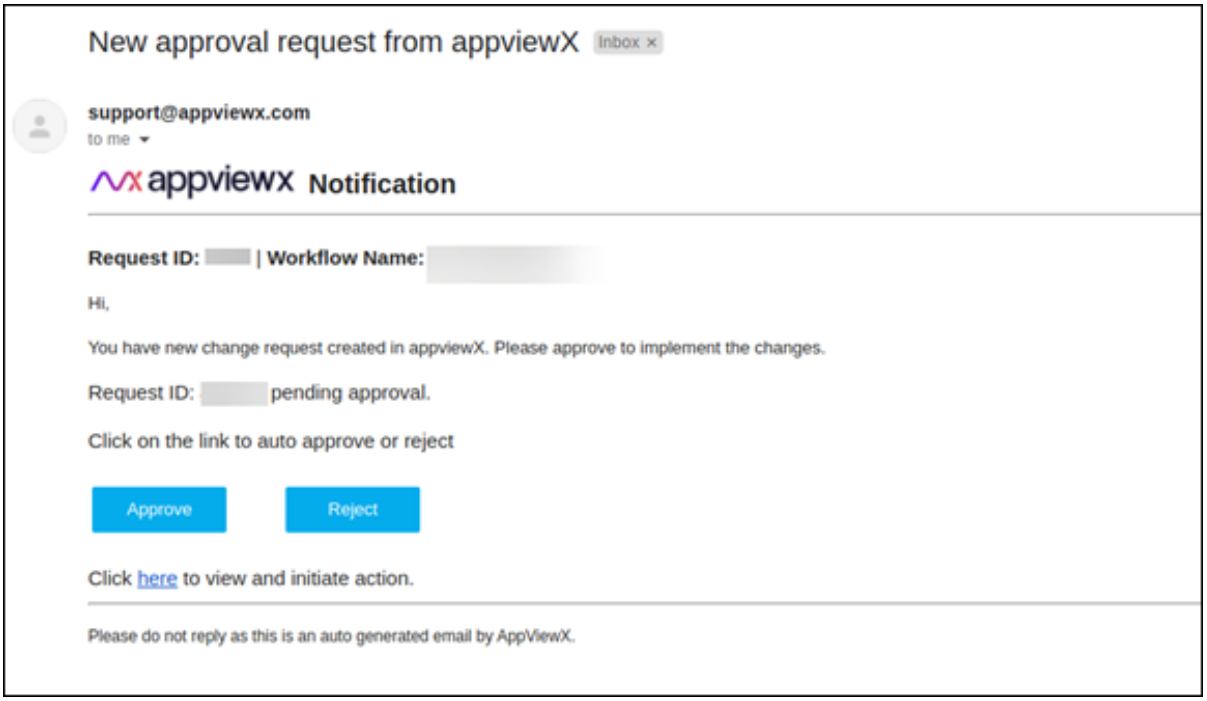
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Click **Preview**.
7. Enter the email ID in the user input form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Approval email notification received via email.

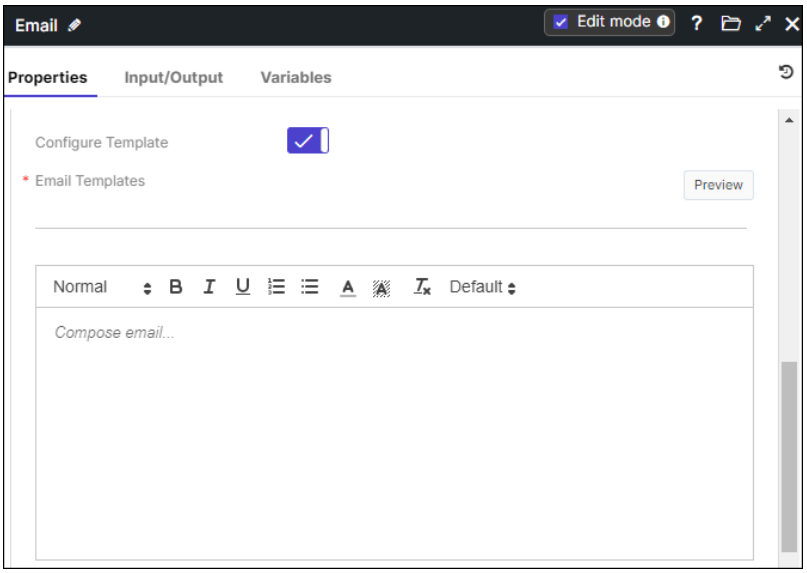


Configuring Email Template

Configuring Template involves selecting and applying pre-configured email templates. This simplifies the content design and formatting process by offering users a variety of elements, including email logos, banners, body text, signatures, and footers, to accompany the email content in the palette.

1. Design a workflow.
2. From the Notificationsection, drag and drop the **Email** task.
3. In the **Email** task window, under **Properties**, in the **General** section, enter the relevant field information and turn on the **Configure Template** toggle.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



On enabling the toggle, ***Email Templates** field appears with the list of available email templates and **Enable custom email header** field disappears.

4. Choose the necessary Email Template from the dropdown, then click **Preview**.
   The **Email Template Preview** popup for the selected email template appears.

> **Note:** For more details on **Creating a New Email Template**, click here.

5. Enter custom email content.
6. Click **Save**.
   You can use this task within your workflow to send email using your customized email template.

> **Note:** If you turn OFF the **Configure Template** toggle button, the current features of the email palette will remain unaltered.

Limitations

- When attaching a file to an email palette using the "attach by file" option, the maximum allowable size is 5 MB, and all file formats are supported.
- For attaching a file via a global variable ("attach by variable"), the maximum size permitted within the form task is 20 MB. Only PDF, CSV, and TXT formats are supported.
- If the configured template is unavailable, an exception will be thrown from the platform.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

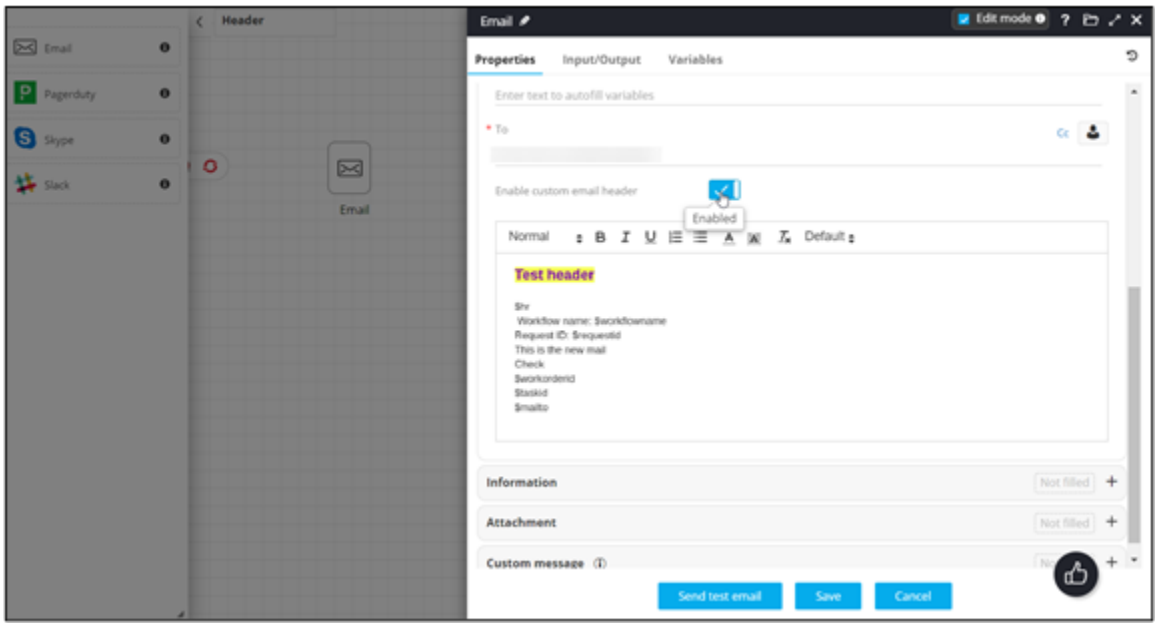
- When using a global variable with HTML content in the body, and if the same content is sent as an email with an email template configured in the platform, users must adjust the width accordingly. Failure to do so may result in misalignment of the logo and footer content from the platform.

  The "Send test mail" functionality and "Auto-reply" functionality remain unchanged.

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.

1. Design a workflow.
2. In the **Email** task, to **Enable custom email header**, turn on the toggle.



3. 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 © 2025 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

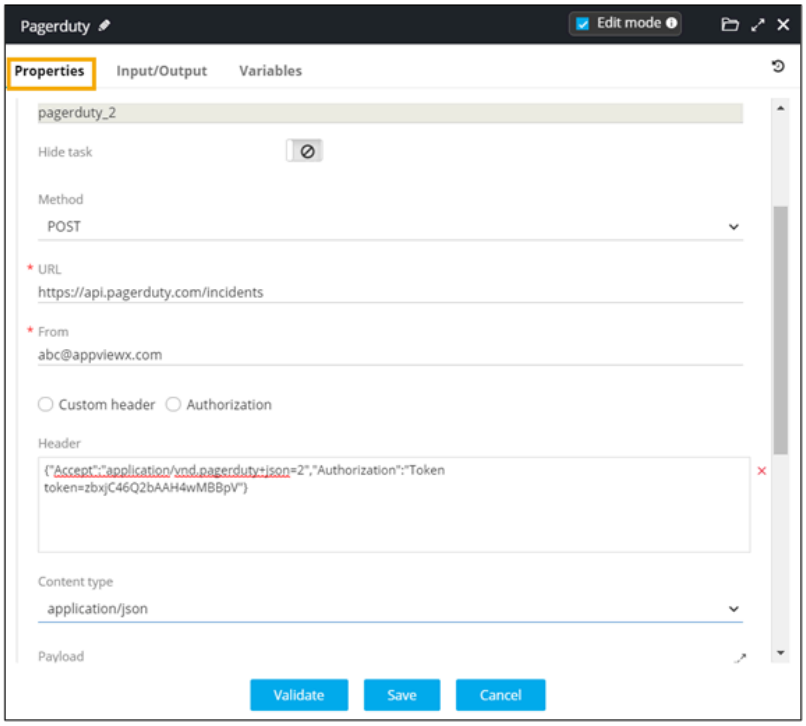
This task allows you to trigger alarm notifications for quick incident remediation through Pagerduty as part of the workflow automation process.

- 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:

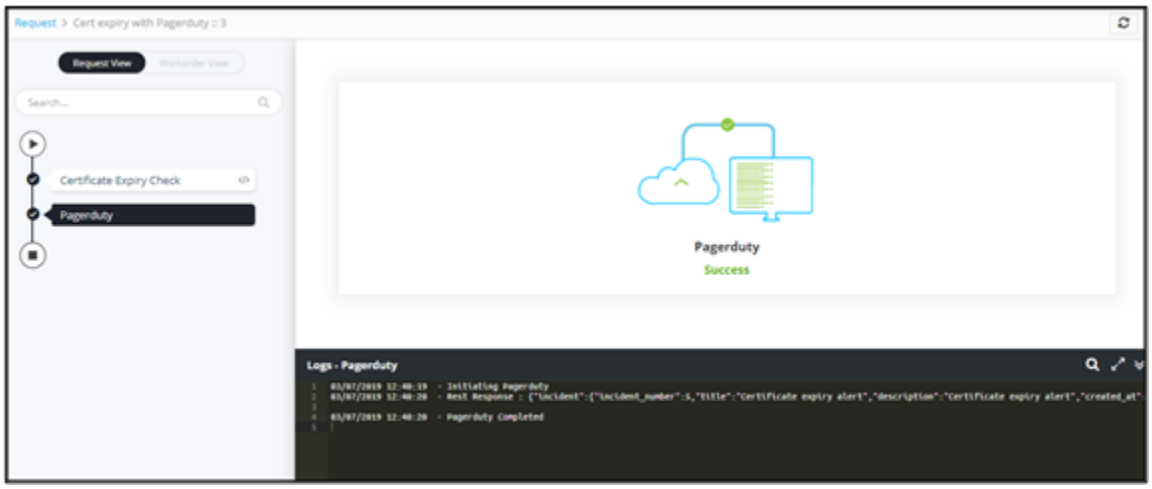
1. Design a workflow.
2. Drag & drop relevant workflow task(s).
3. Double click on the **Pagerduty** task.
4. In the **Pagerduty** task window, under **Properties**, enter or select the field information as shown.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



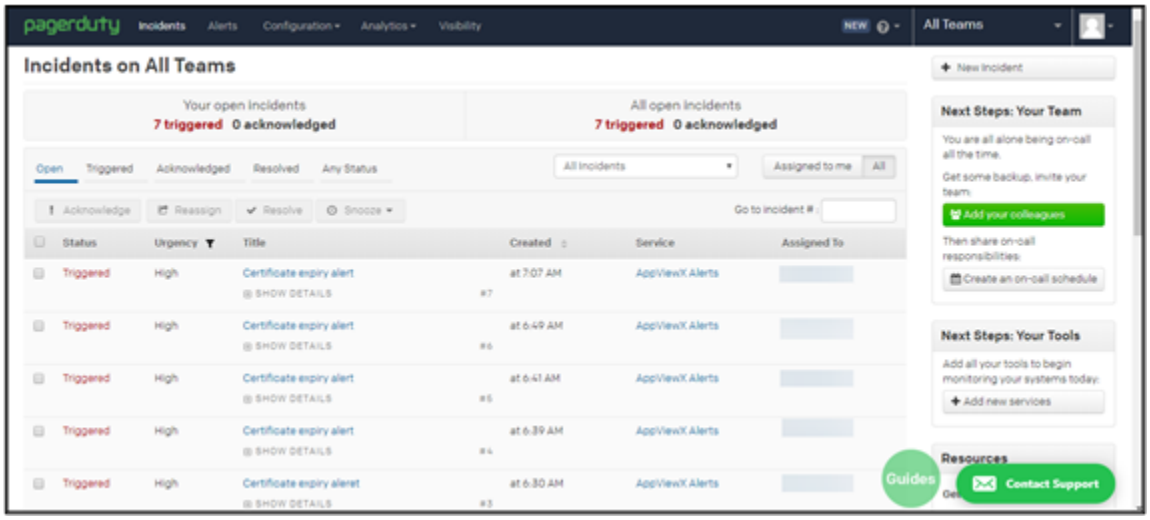
5. Enter the intended payload, with global variable(s) if any.
6. Connect the workflow tasks and enable the workflow.
7. Trigger the workflow from the Request :: View/Run page.

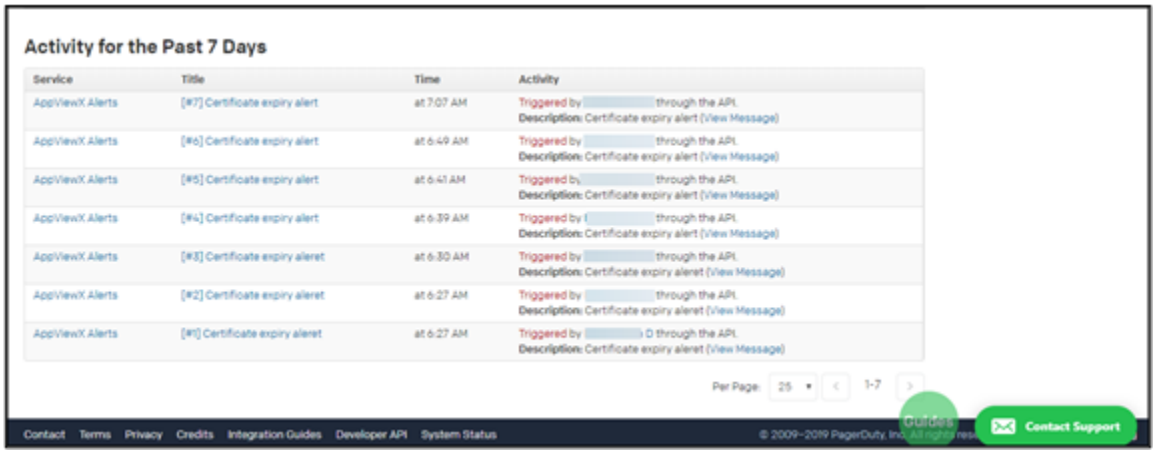
   Workflow is executed successfully.



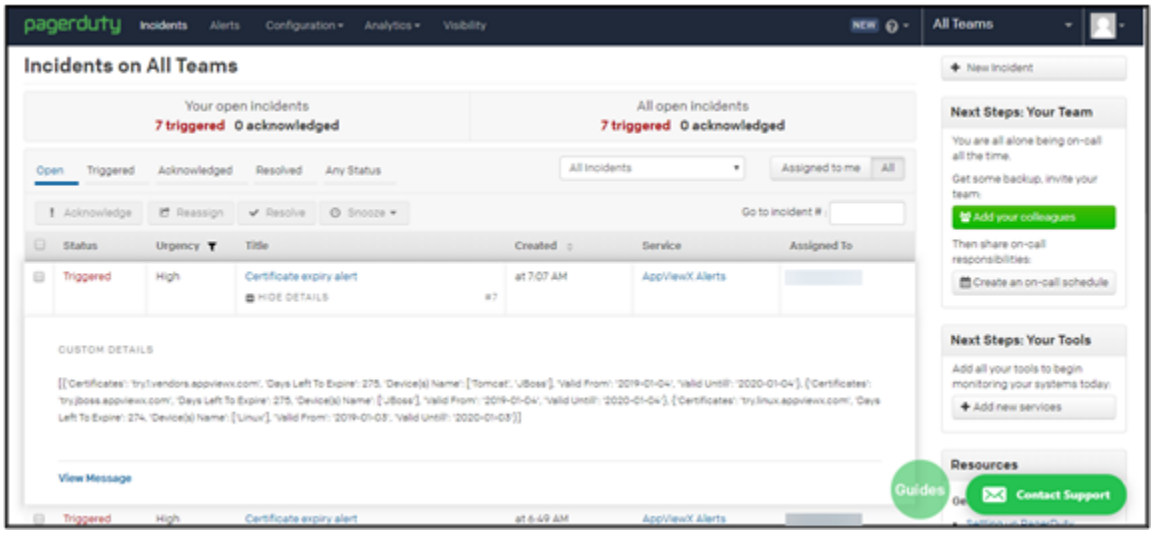
8. To view all Incidents on Pagerduty, log into your account using valid credentials.

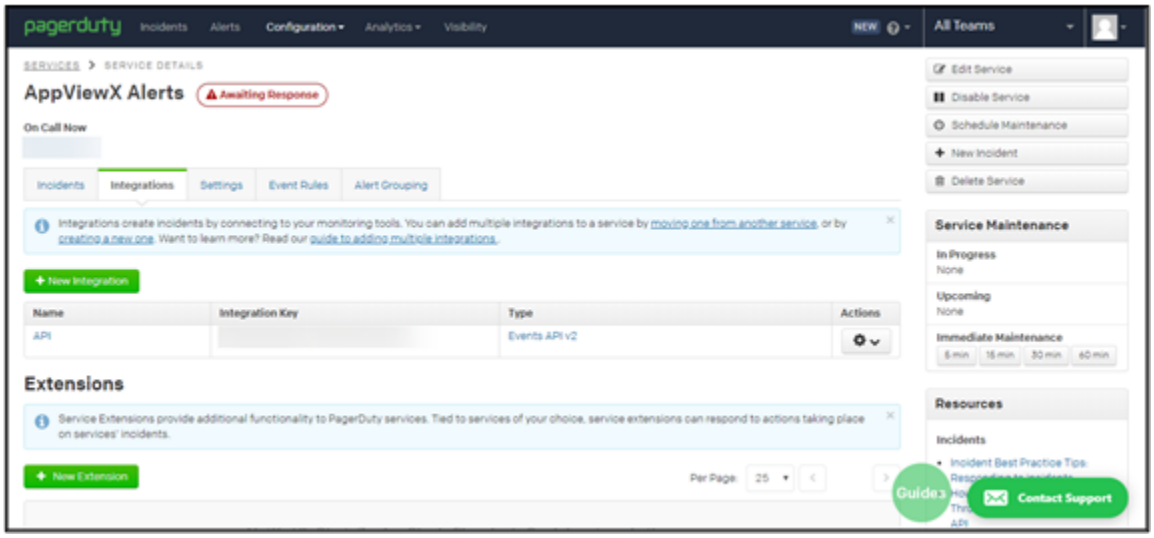
Copyright © 2025 AppViewX, Inc. All Rights Reserved.





Copyright © 2025 AppViewX, Inc. All Rights Reserved.





Slack

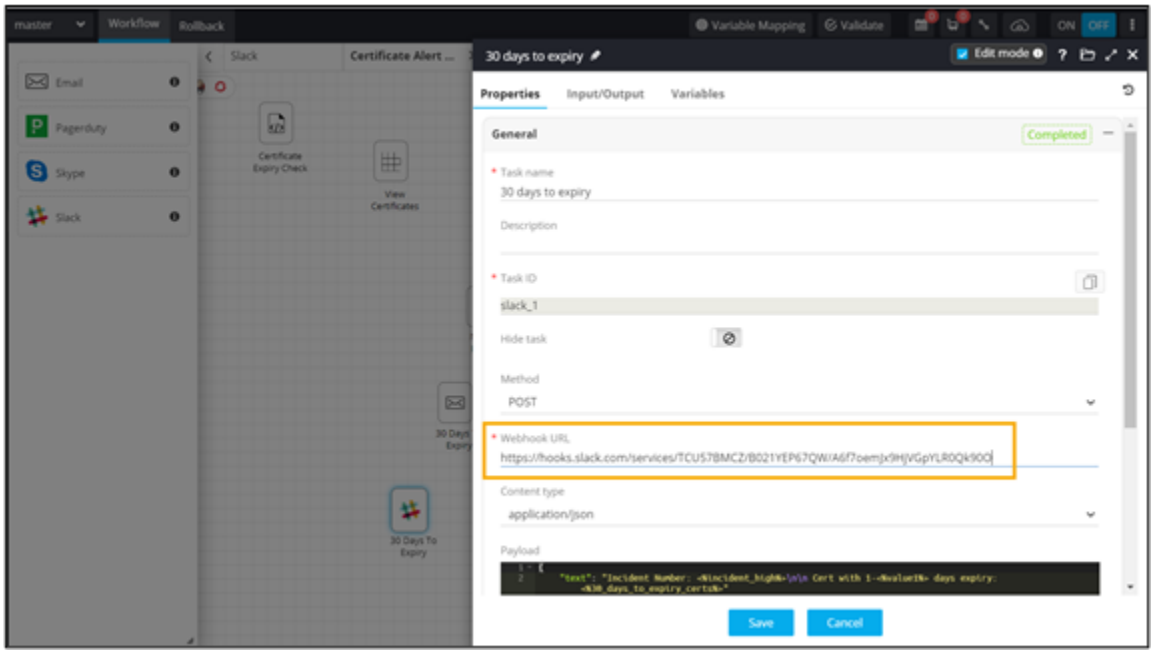
This task allows you to send notifications on a Slack channel as part of the workflow process.

- 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.

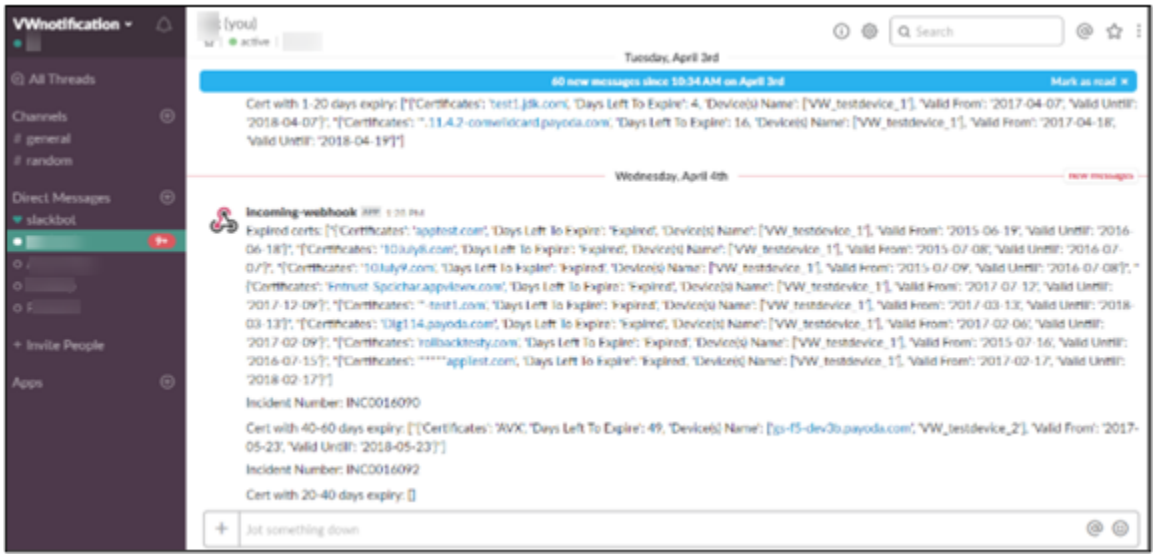
To send a notification through Slack:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. Drag and drop relevant workflow tasks.
3. From the Notification section, drag and drop the **Slack** task.
4. Define the appropriate webhook.



5. Enter the intended payload with global variables, if any.
6. Connect the workflow tasks and enable the workflow.
7. Trigger the workflow.
8. View Slack notifications.

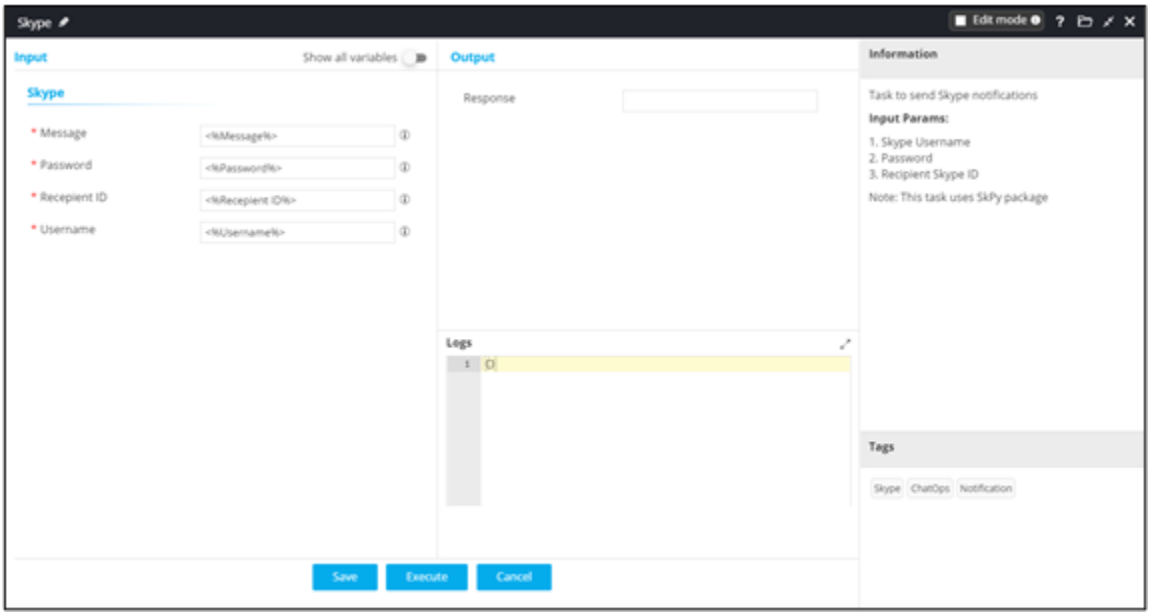


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Skype

This task allows you to send Skype notifications as part of the workflow process.

- 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.



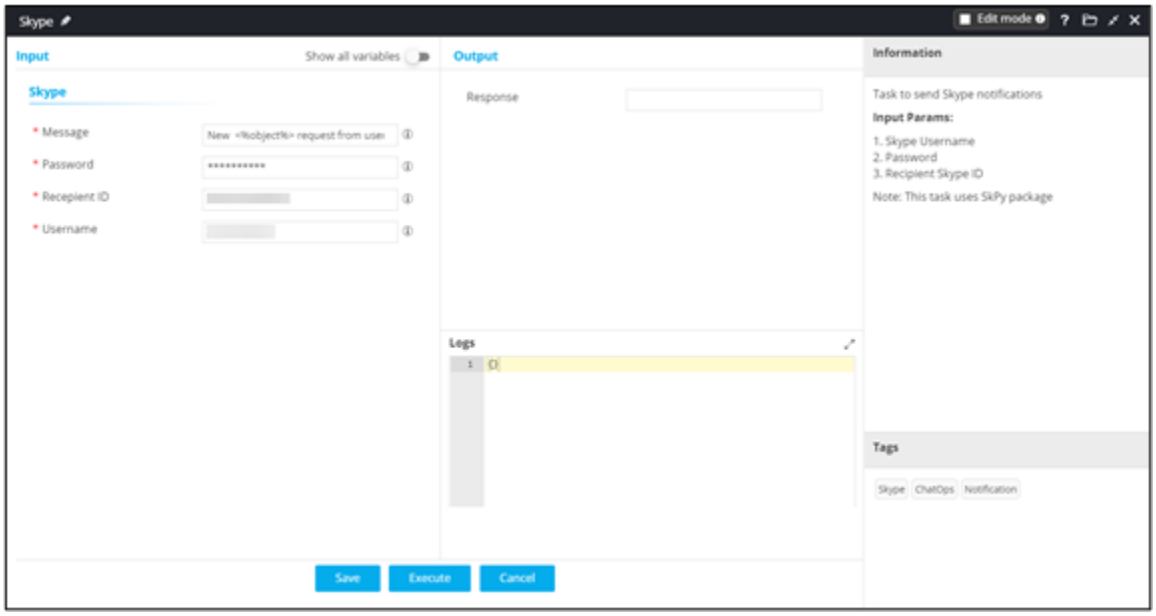
To send a notification through Skype:

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.

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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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 the workflow tasks and enable the workflow.
7. Trigger the workflow 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.

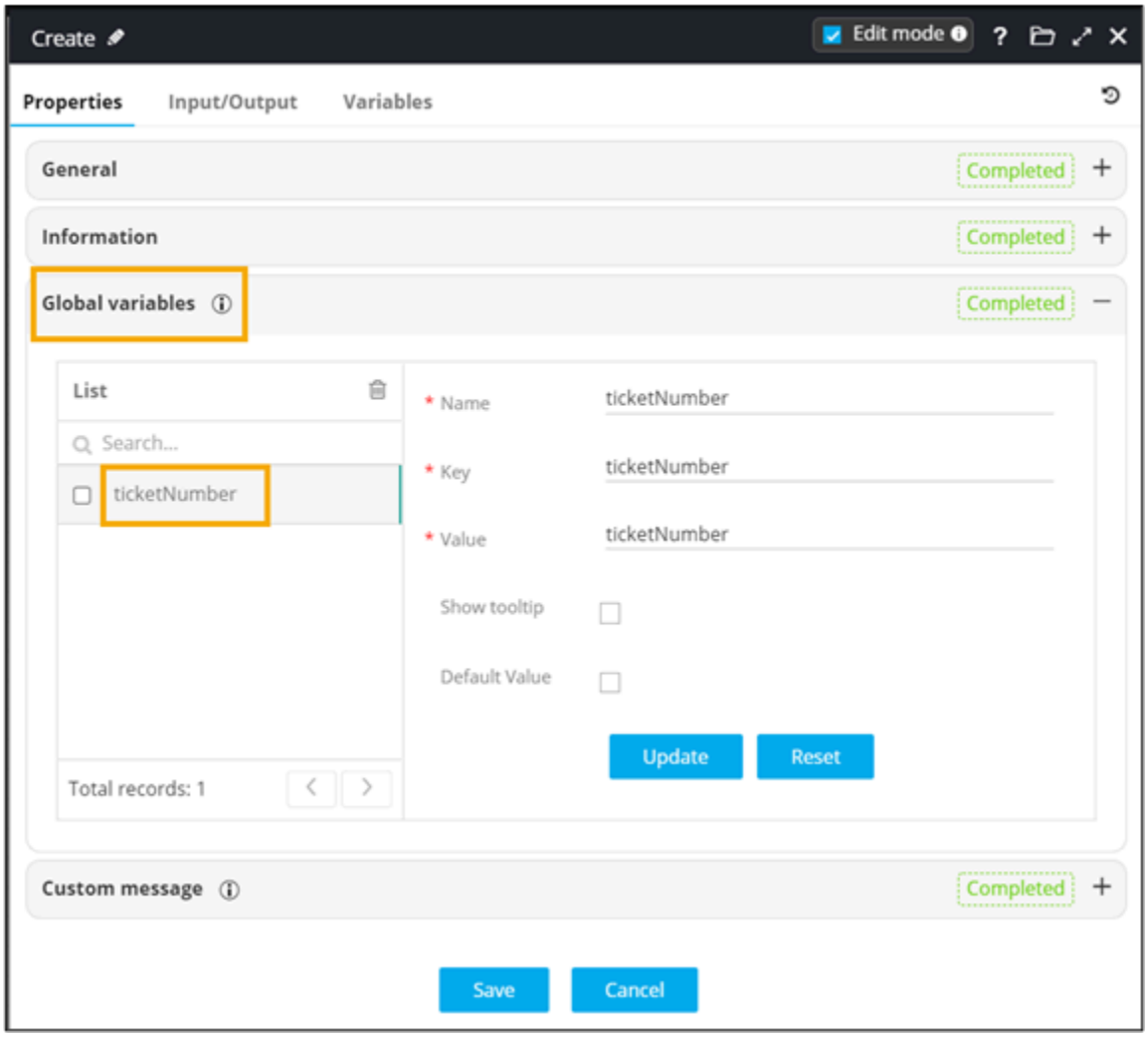
To create a workflow with ServiceNow tasks:

1. Design a workflow.
2. In the design workspace, under the **Change Management** folder in the left menu, from the **CHANGE** folder, drag and drop the **Create** task.

> **Note:** Timestamp must be in milliseconds.

3. In the **Create** task window, under **Properties**, in the **Global variables** section, declare the necessary global variables.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. From the **CHANGE** folder, drag and drop the **Get** task.
5. In the **Get** task window, under **Properties**, pass the global variable declared in the previous task.
6. Connect the workflow tasks and enable the workflow.
7. Trigger the workflow.

   Workflow is executed successfully.

Provisioning for Workflow Tasks

You can define specific configurations for executing the provisioning workflow tasks such as Implementation, Prevalidation, Postvalidation, and Rollback.

To define the configurations:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

New Menu	Old Menu
1. Form the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, click **WORKFLOW** > **Provisioning**.	1. Form the main navigation menu, select **Settings**. 2. On the **Settings** module, from the left menu, click **Provisioning**.
Note: For more information on how to switch between menus, click here.	

1. Under the **Work order configuration** section, define the configurations for Both or Manual or Auto of the **Implementation type**.
2. In the **Implementation type** field, select the Implementation type from the dropdown.
3. Under the **Command configuration** section, define the configurations for success or failure of the implementation task.
4. In the **Failure messages** field, enter the keywords for error responses to identify the task as failed as shown in the image below.

   If the workflow execution displays any of the keywords defined above as error messages, the task will be considered as failed.

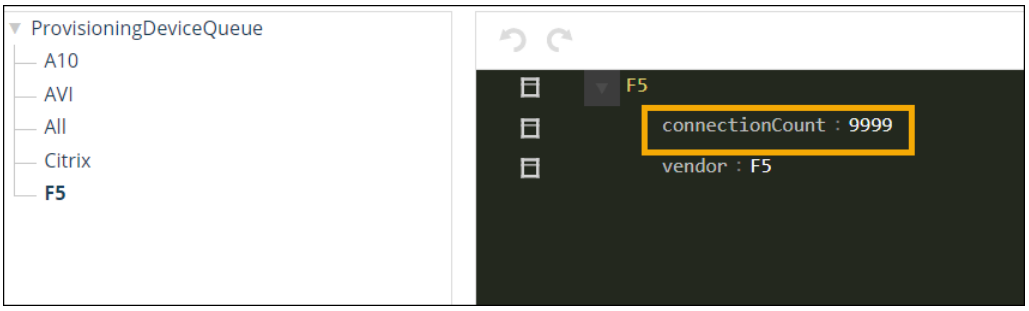
5. In the **Excluded failure messages** field, enter the keywords to be excluded for identifying the task as failed.
6. In the **Commands to wait** field, you can enter a set of commands and define a time out for the implementation task. If the task contains the configured commands, then it will time out as per the time out defined. This is useful for tasks or commands that take longer than the default time to execute.
7. In the **Time out (mins)** field, enter the timeout value for the tasks having the configured set of commands.

   Note: The default value is 0.5 minutes or 30 seconds.

8. Turn on the **Continue on a command failure** toggle, to proceed in case of a command failure.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

9. Turn on the **Enable Device Queue** toggle to enable queuing of device connections as per the defined connection count in the Collection.



10. Turn on the **Enable Device Session Cache** toggle, to check for any existing open sessions for the device and execute other requests in the same session.
11. Turn on the **Rollback work order** toggle to rollback configurations that are partially implemented.

> **Note:** Configurations that were successfully implemented will be excluded from the rollback work order.

12. Under the **Rest configuration** section, define the different success status codes in the **Rest success status code** field.
13. Under the **Collection configuration** section, turn on the **Enable default collection** toggle to create a collection in the name of the device and store all the IPs in the device (using a predefined parsing logic) during the next device parsing.

Connecting Workflow Tasks

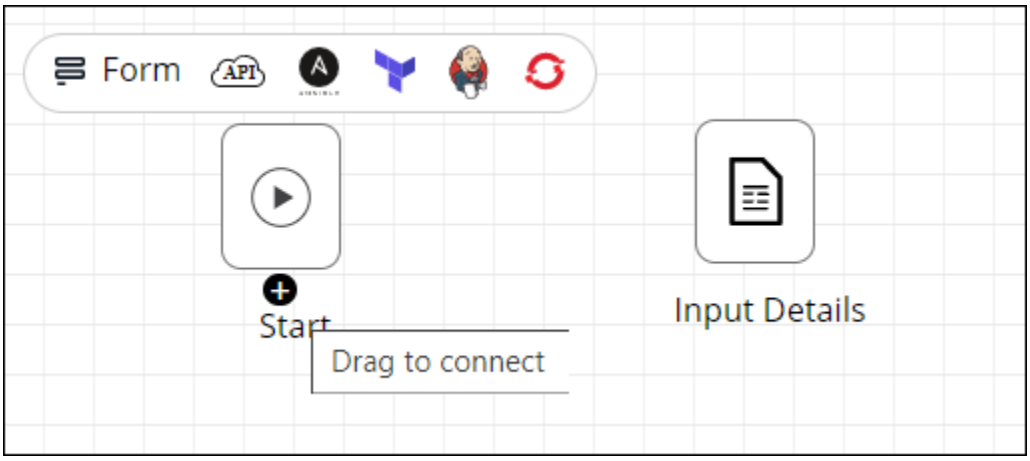
You cannot trigger a workflow until all the tasks are connected.

To connect the tasks:

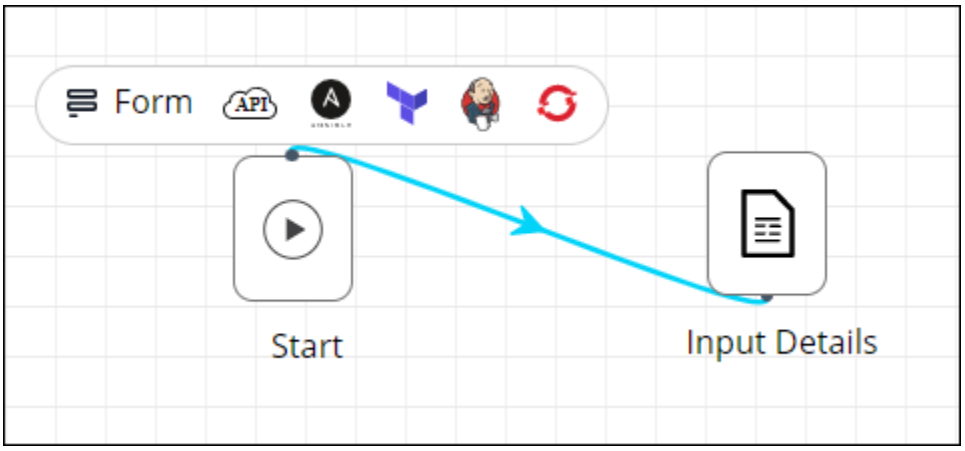
1. Hover your mouse over the first task.

   A **(Start)** icon displays under it.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. Place the cursor over the **(Start)** icon and drag your mouse to connect to the next task.
   The tasks are connected.



- Customizable Connectors
- Decisions via Connectors/Links
- Connecting Workflow Tasks with Failover - RGF Flow

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:

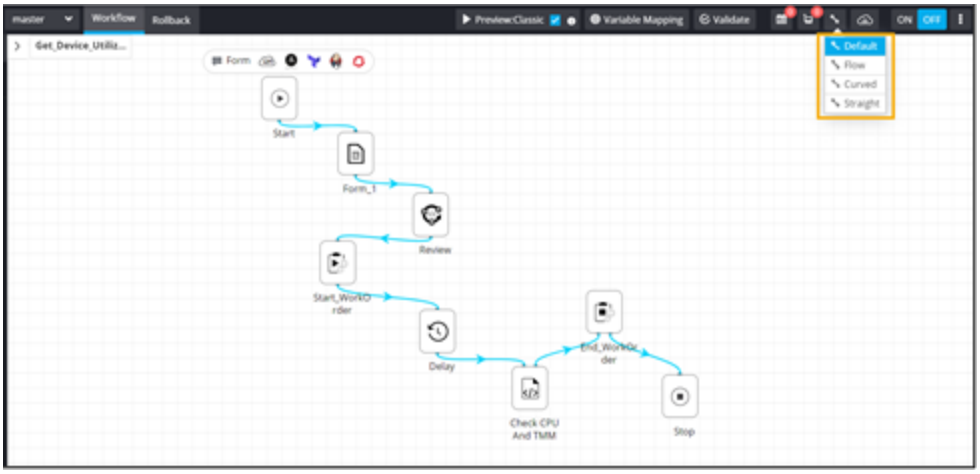
- Default
- Flow

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Curved
- Straight.

To customize connectors:

1. Go to the Workflow page.
2. Open an existing workflow.
3. In the design workspace, to view the different connector options, click **(Default)** icon.
4. Select the connector from the available options.



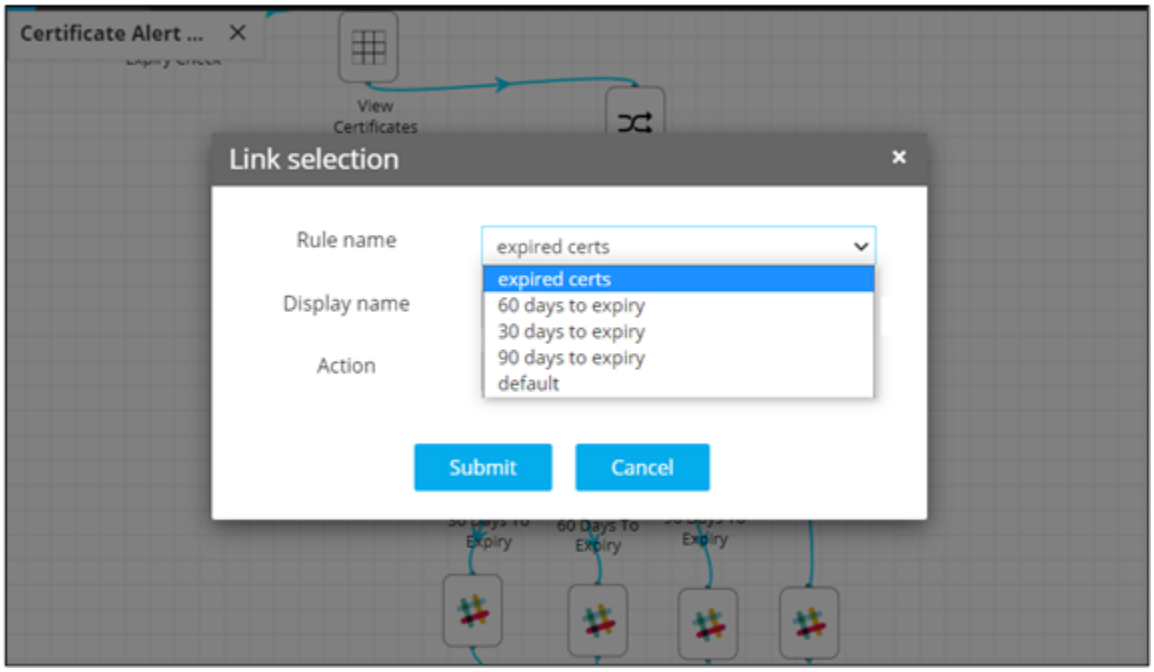
Decisions via Connectors/Links

Connectors or Links allow you 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 © 2025 AppViewX, Inc. All Rights Reserved.



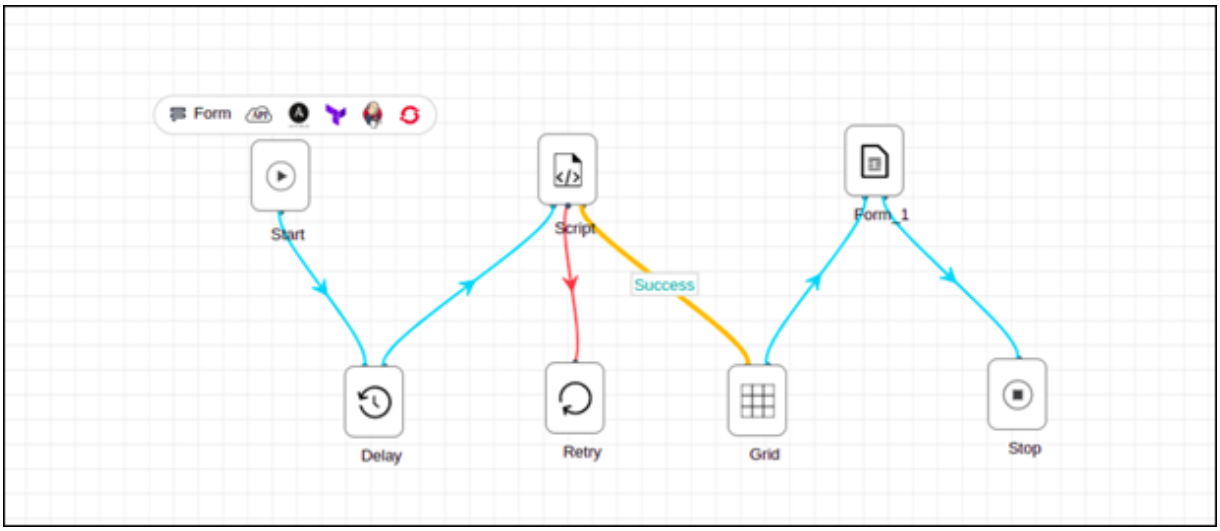
Connecting Workflow Tasks with Failover - RGF Flow

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.
4. In case of success, proceed to the next task.

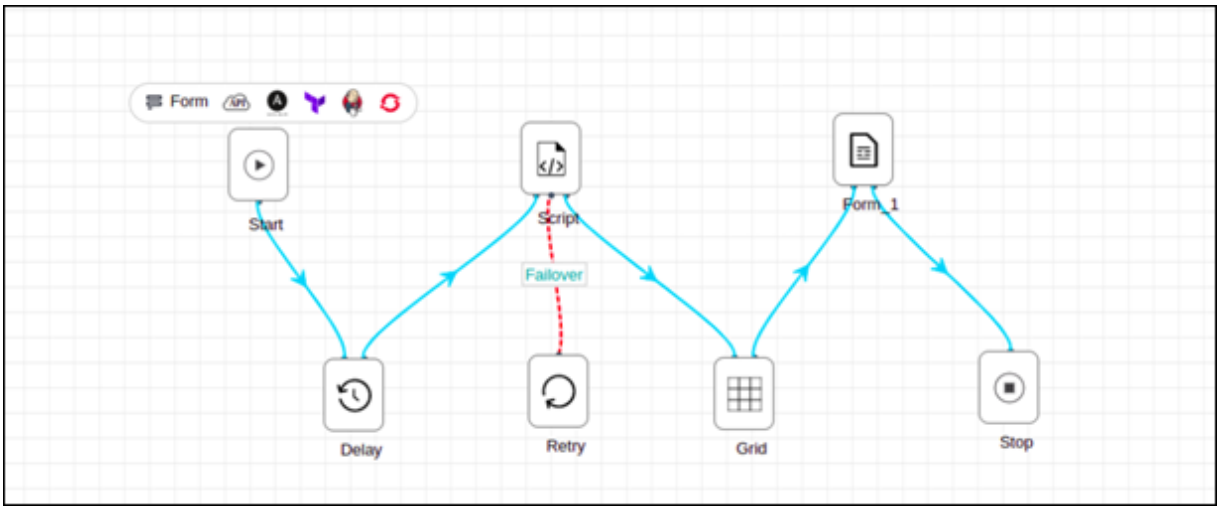
   The **Success** link is displayed in blue color.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



5. In case of failover, send an email notification and perform a retry.

   The **Failover** link is displayed in red color.



Note: The selected link is displayed in orange color.

Adding Variables to Tasks

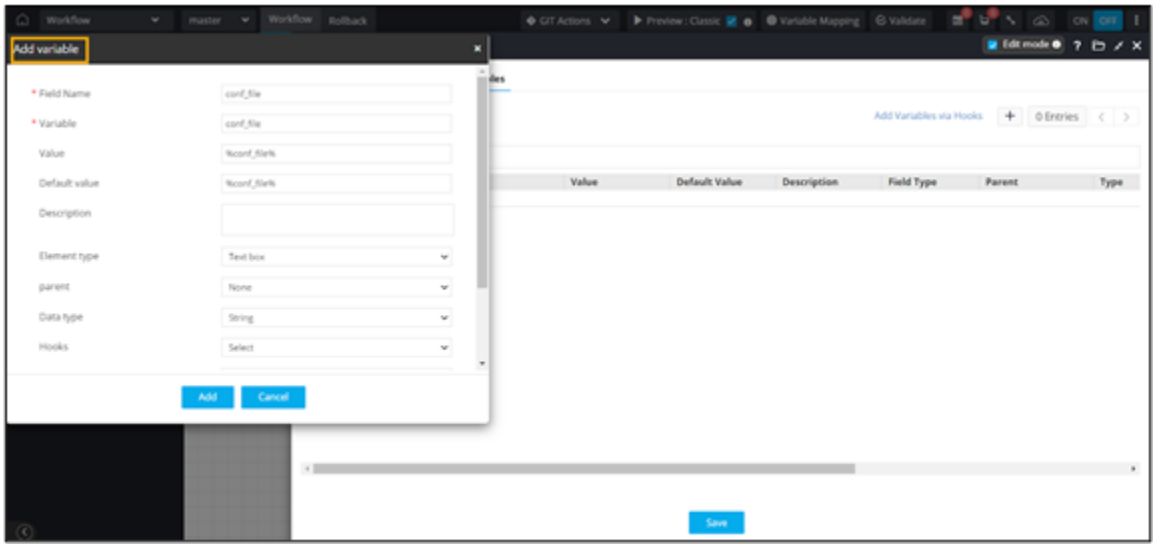
This feature allows you to add variables to tasks to support auto-generation of form in the workflow.

To add variables to a task:

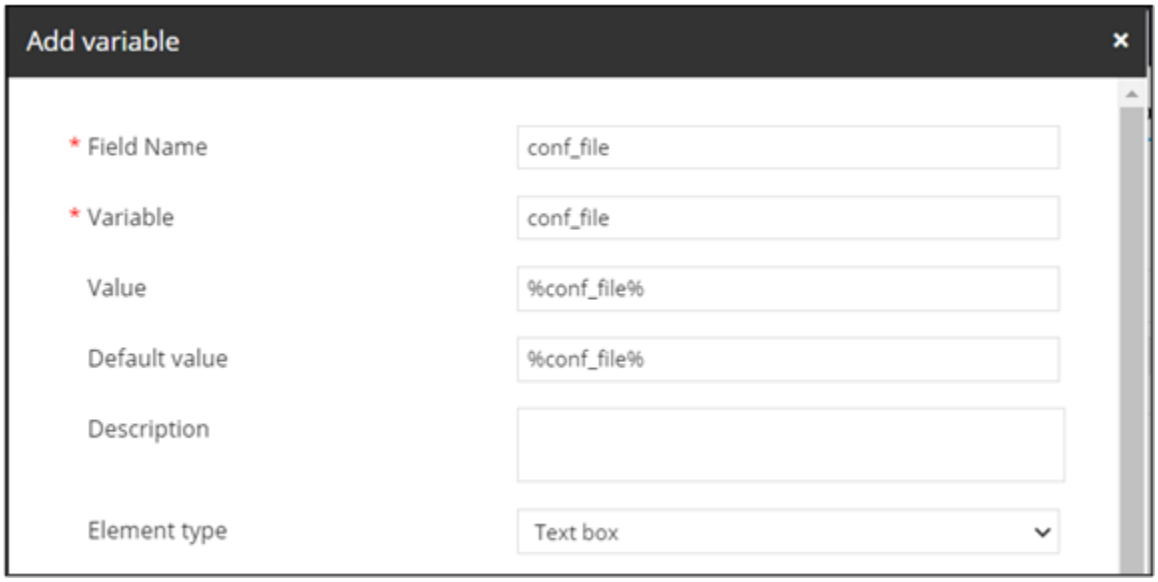
1. Double-click on any task.
2. In the task window, click **Variables**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

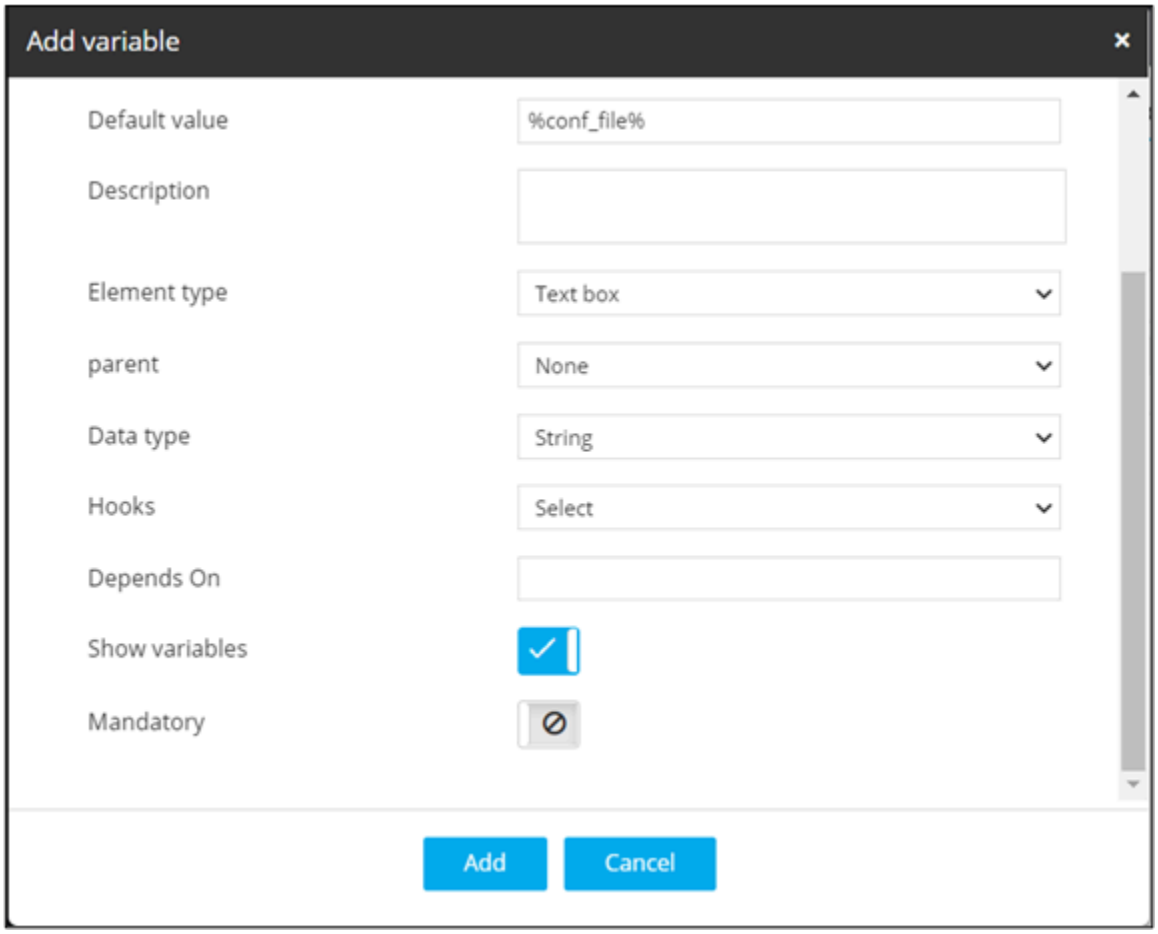
3. Under the **Variables** tab, to add a variable, click **(Add)** icon.
   The **Add variable** window is displayed.



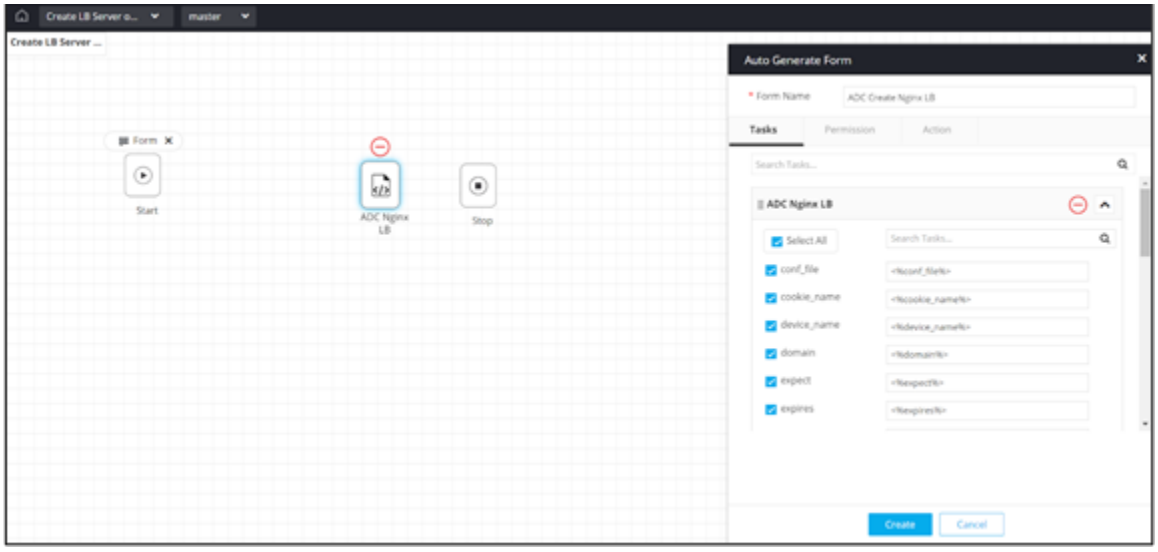
4. In the **Add variable** window, enter or select the field information based on the variable that needs to be added.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

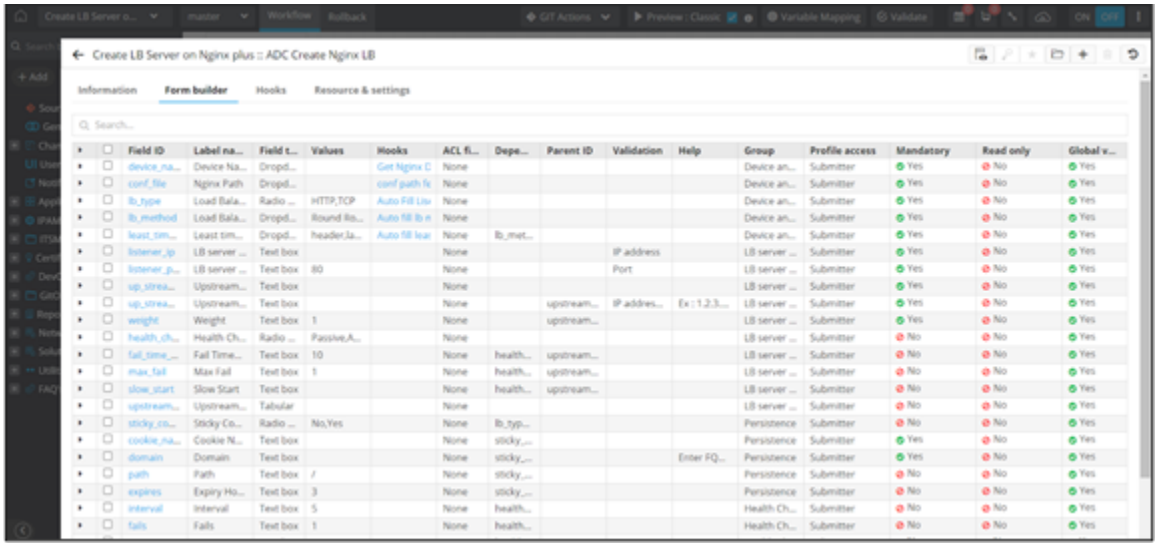


5. Click **Add**.
6. Once you have added all the variables, click **Save**.
7. Auto-generate a self-service form based on the variables defined in the Script task.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The form is auto-generated based on the variables defined in the Script task.



8. Connect the workflow tasks and enable the workflow.
9. Trigger the workflow.

Automation Collision

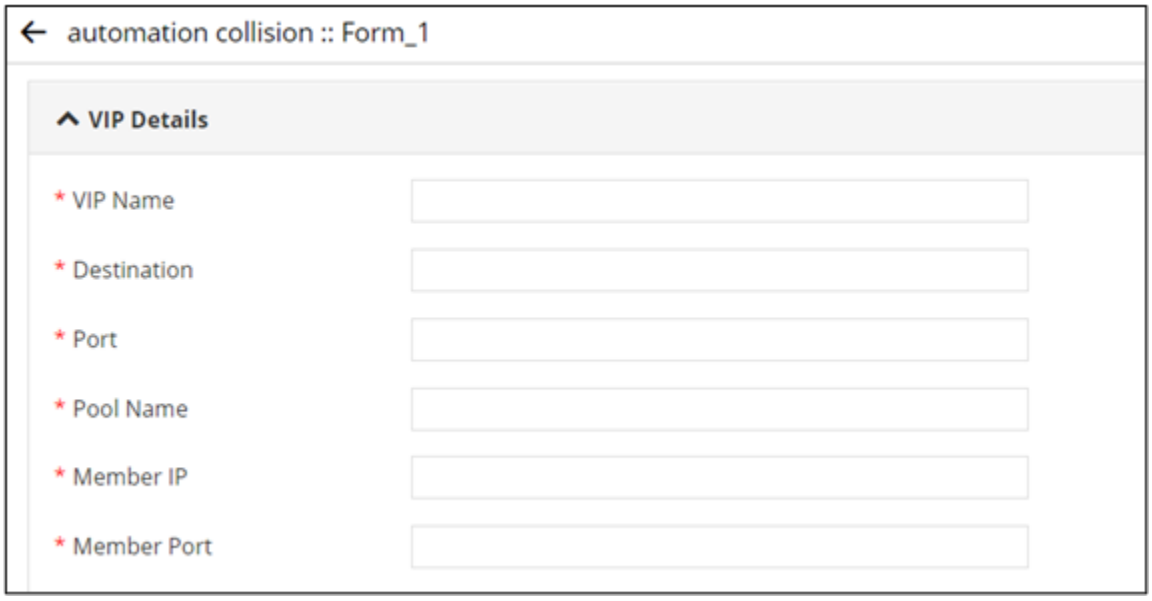
This feature allows you to check for duplicate configurations across workflow requests.

- 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 configuration validation.

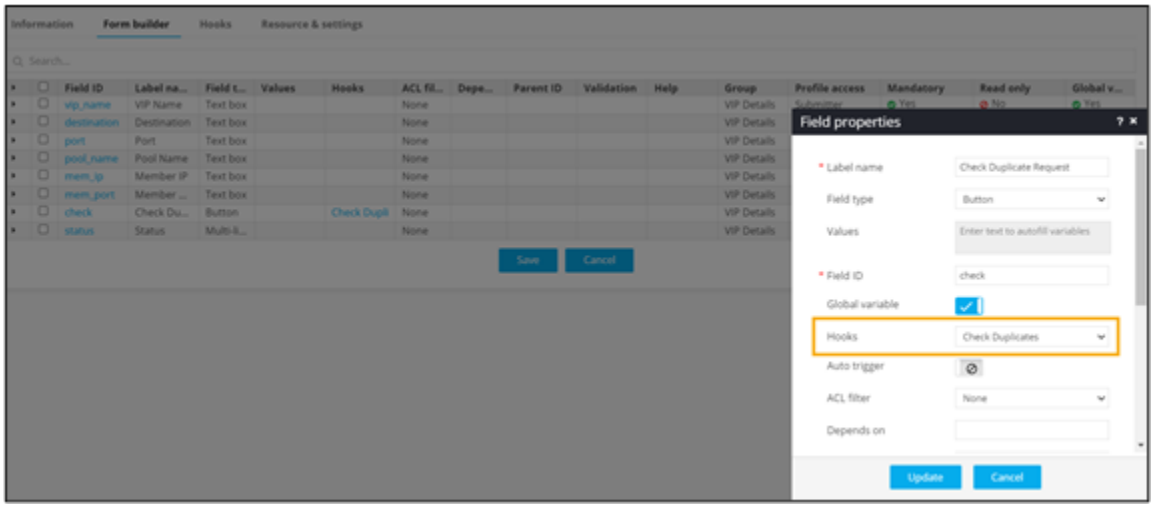
To check for duplicate configurations when designing a workflow to create a virtual server:

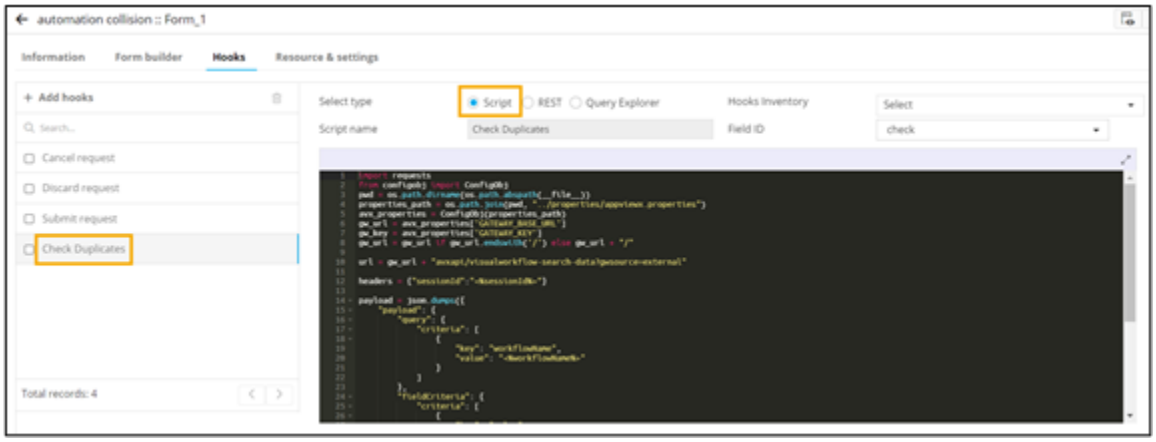
1. Design a workflow.
2. From the User Interface section, drag and drop a **Form** task.
3. Add the form fields for creating a virtual server.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. Add a button field to check for duplicate configurations using a Script logic.





Copyright © 2025 AppViewX, Inc. All Rights Reserved.

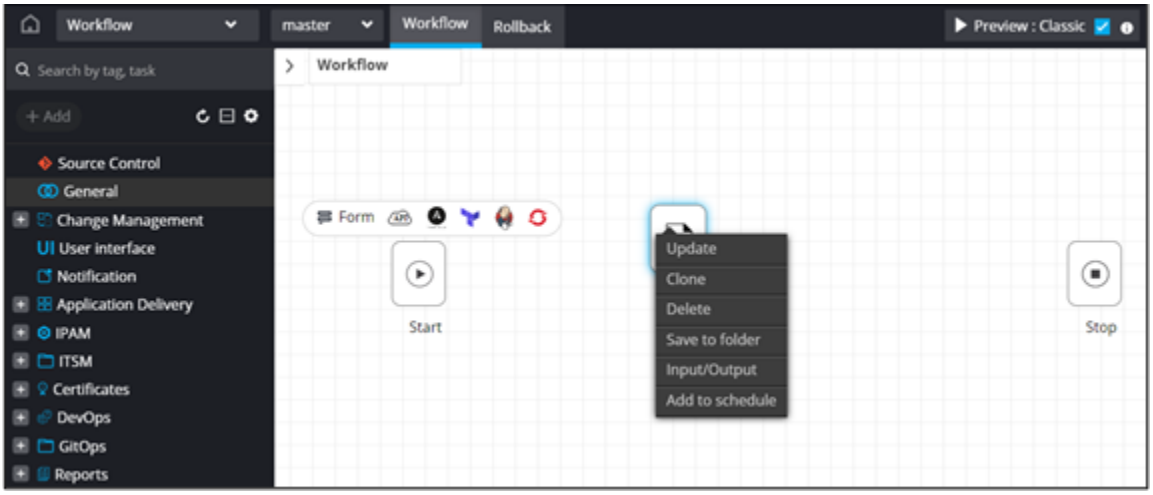
5. Connect the workflow tasks and enable the workflow.
6. Trigger the workflow.

   Workflow is executed successfully.

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.

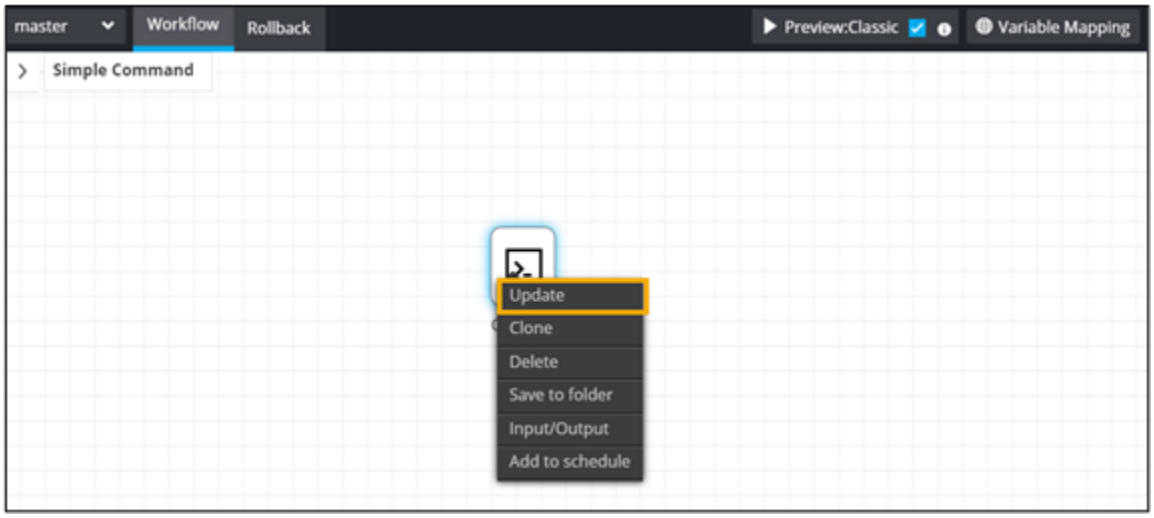


- Updating a Task
- Cloning a Task
- Deleting a Task
- Saving and Reusing a Task
- Input/Output Variable Mapping

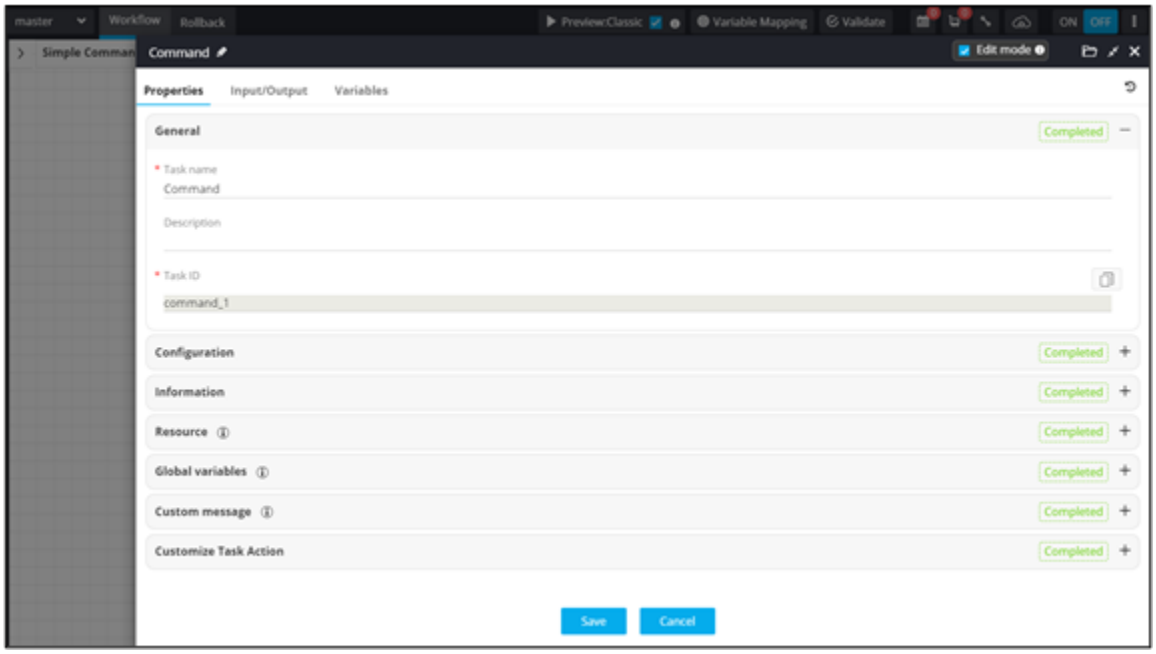
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Updating a Task

1. Right-click the task to see options.
2. Click **Update**.



3. Edit the task properties to update the task.

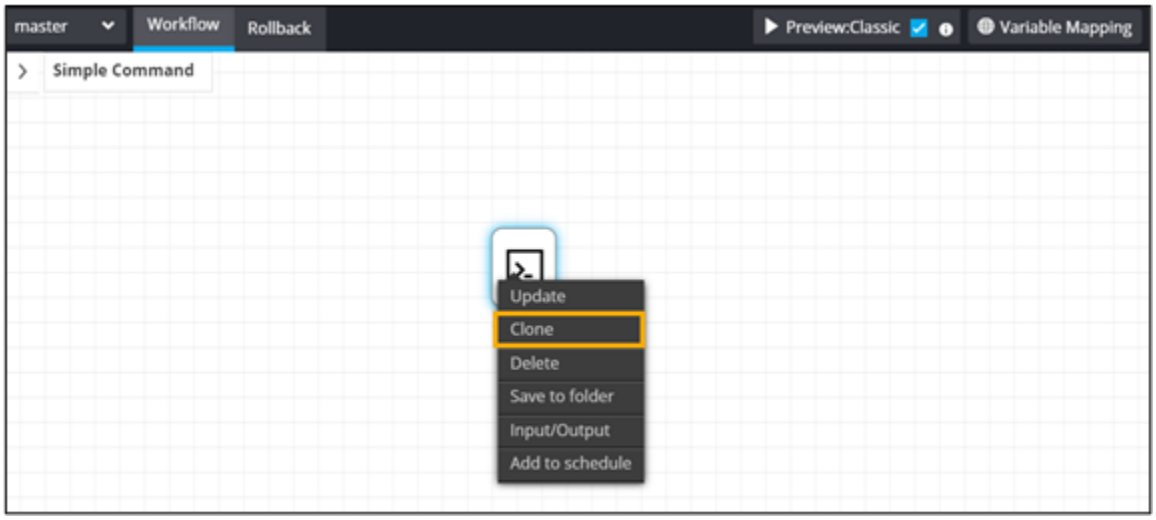


Cloning a Task

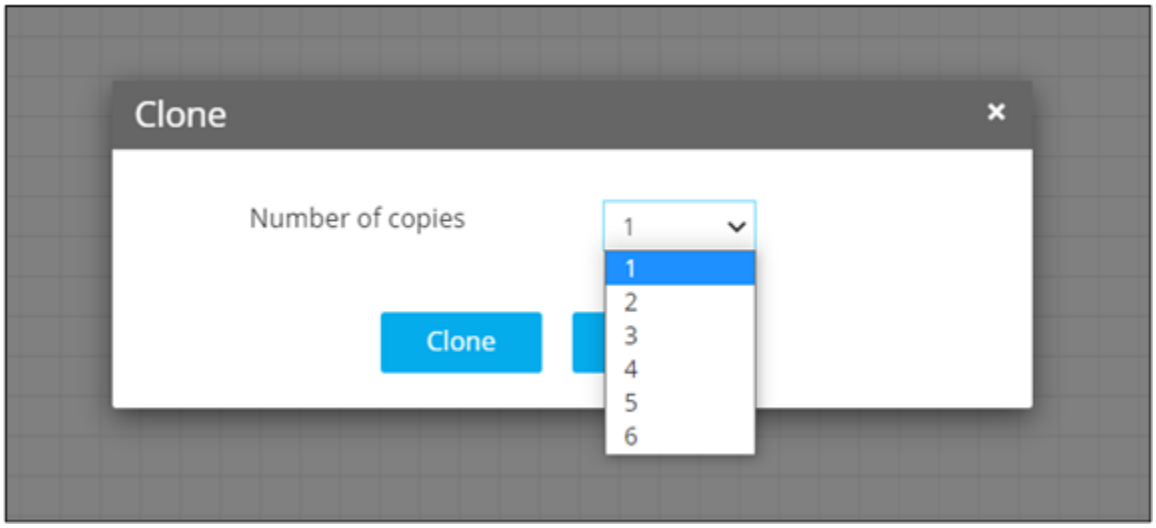
You can clone a task in the workspace instead of having to drag and drop the same task from the menu.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Right-click a task to see the options.
2. Click **Clone**.



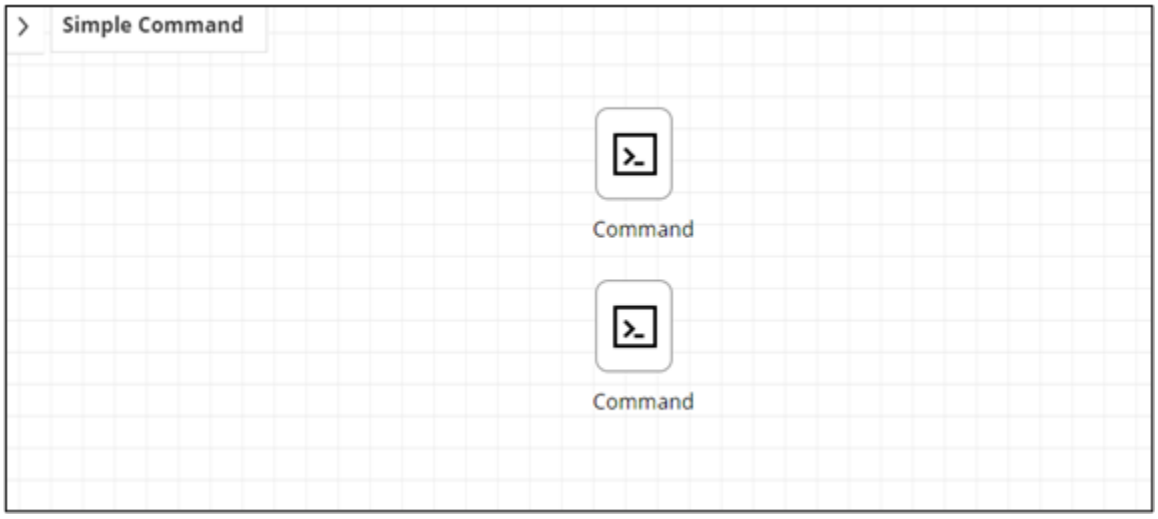
3. In the **Clone** window, select the number of copies from the dropdown list.



4. Click **Clone**.

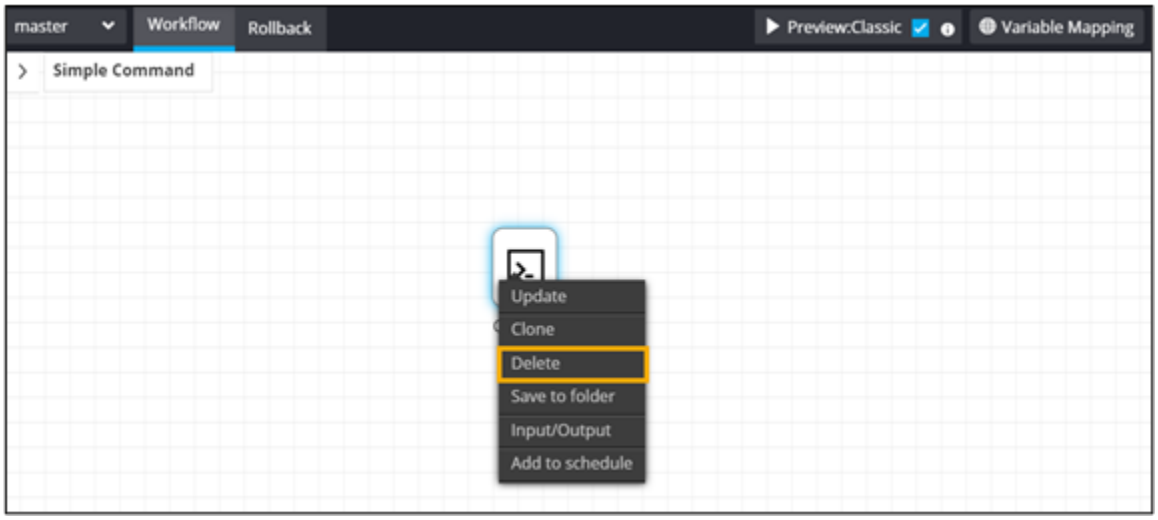
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A copy of the task is created in the workspace.



Deleting a Task

1. Right-click on the task to view options.
2. Click **Delete**.



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

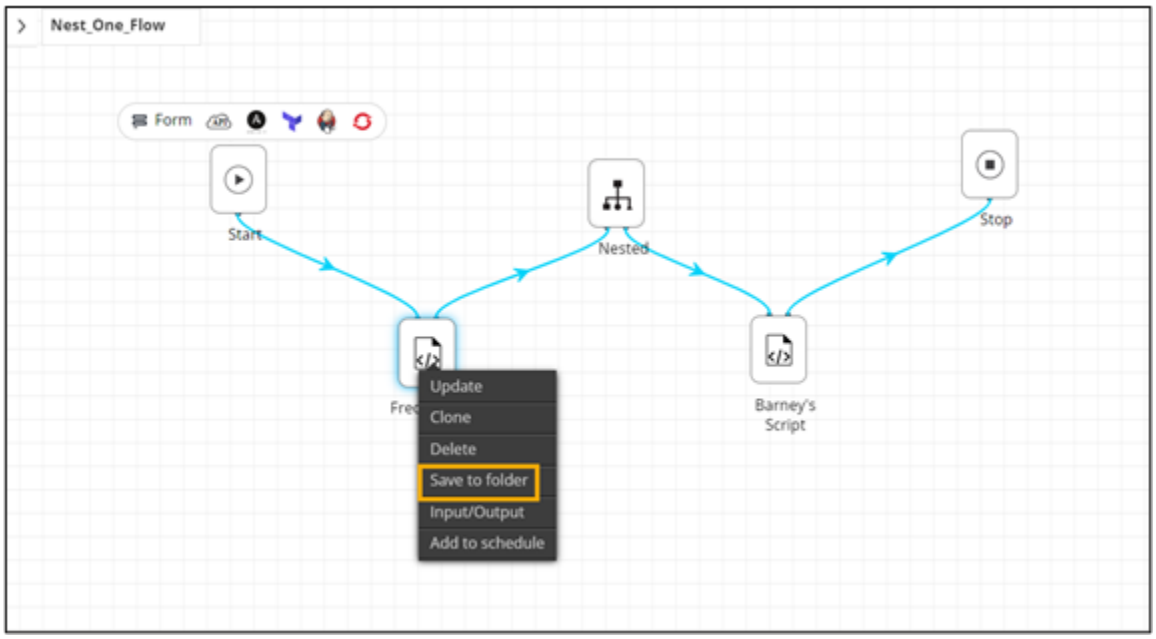
   The task is deleted.

Saving and Reusing a Task

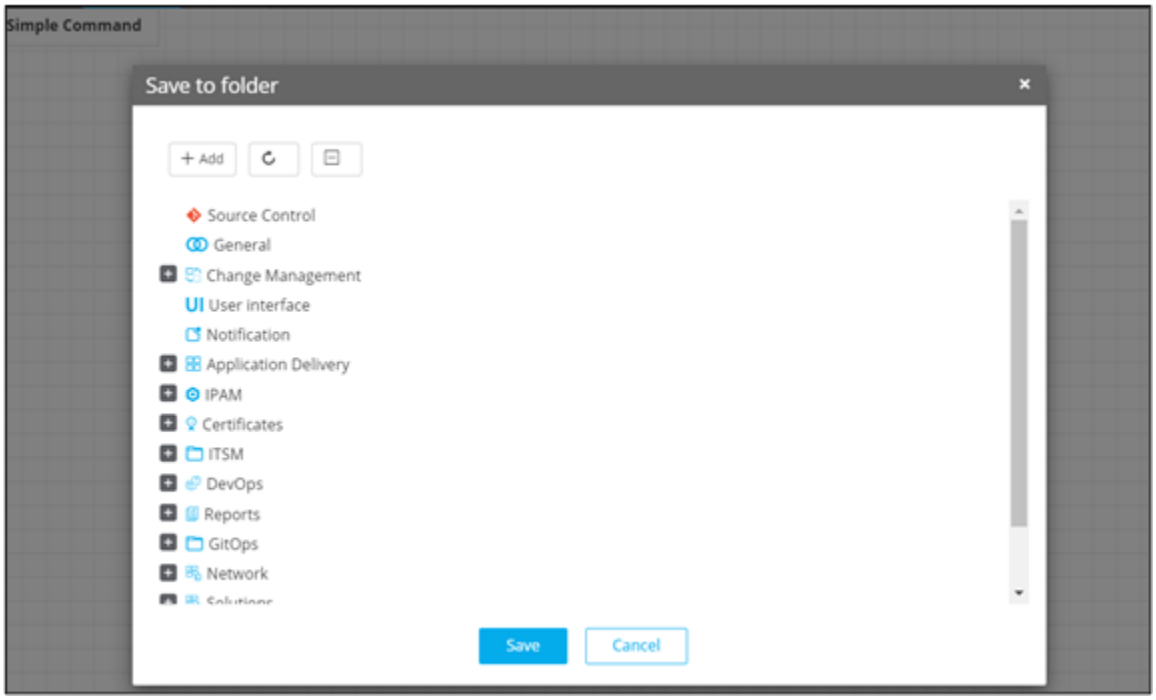
You can save specific tasks to a folder(s) in order to import/reuse them in other workflows.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Open an existing workflow in the Workflow Studio.
2. Right-click on the task to be saved.
3. From the options displayed, select **Save to folder**.



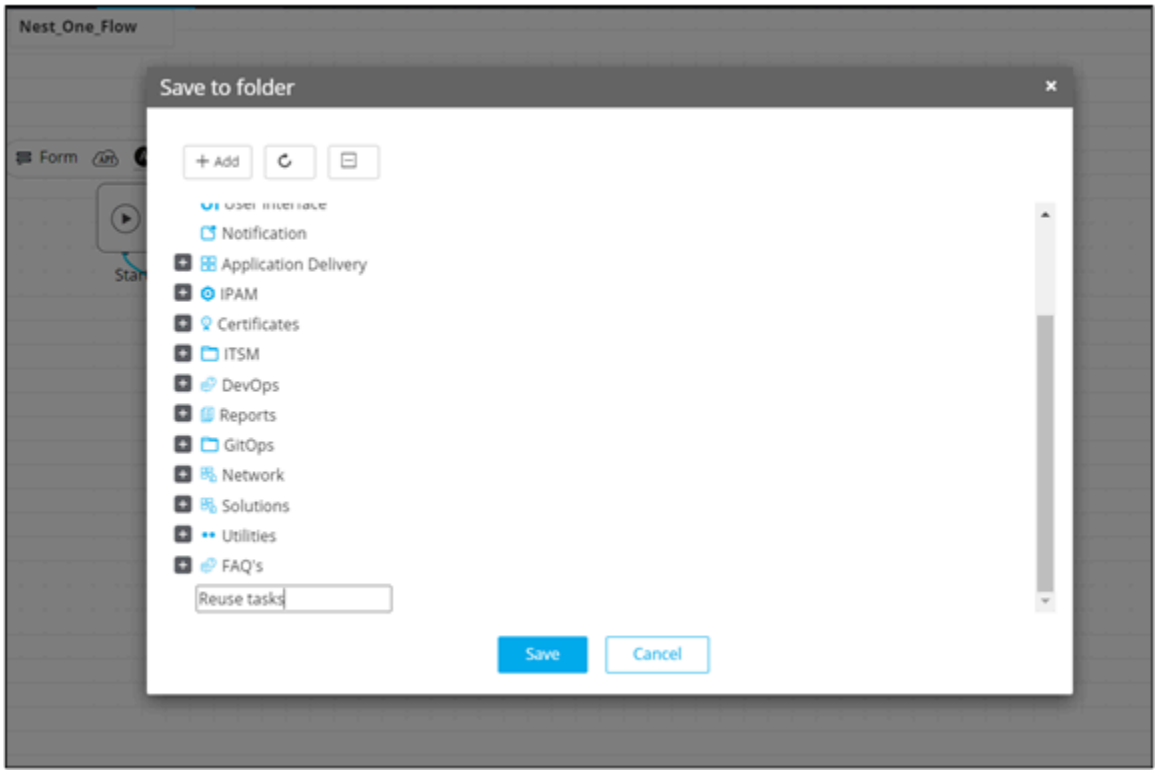
4. To save the task to an existing folder, in the **Save to folder** window, select a folder from the list.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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**.

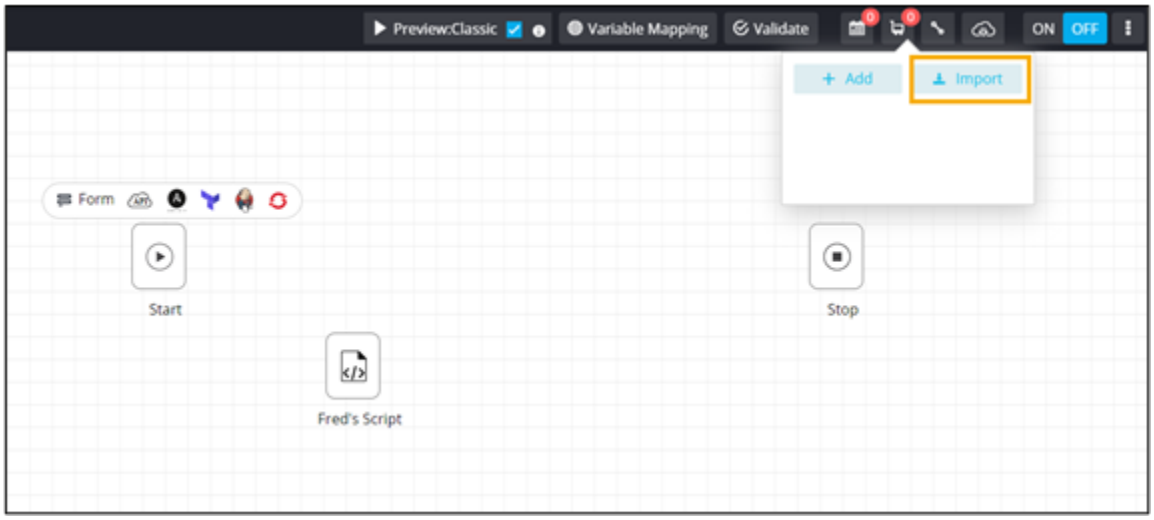
5. Click **Save**.
6. To save the task to a new folder, click **Add**.
7. Enter the folder name and click **Save**.



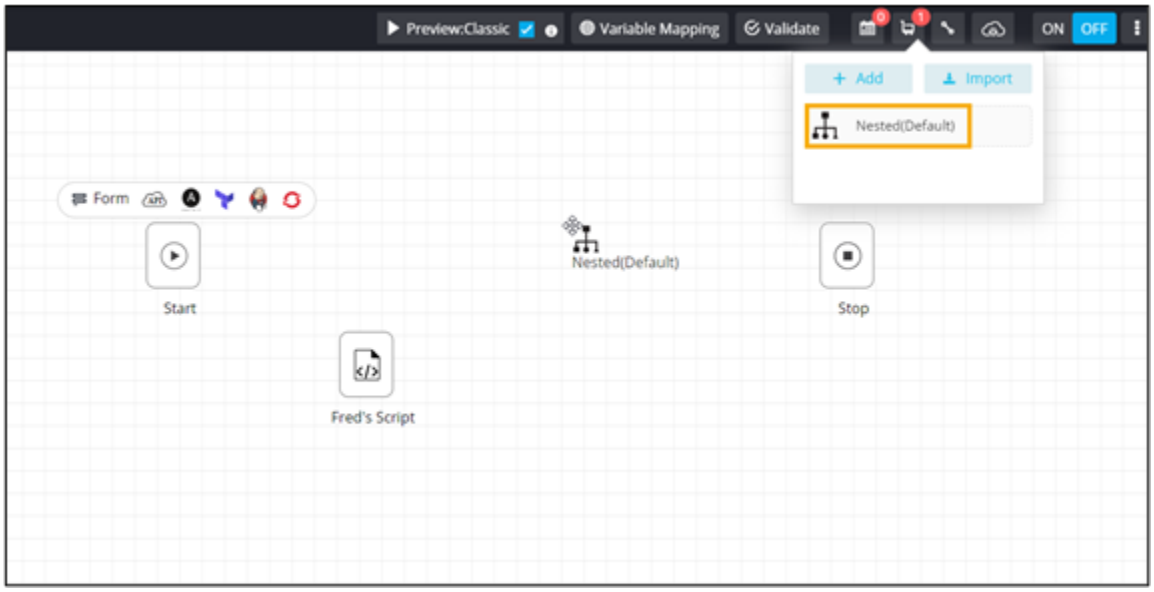
The new folder is added to the left menu.

8. Exit the workflow.
9. Design a new workflow.
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 **(Cart)** icon.
12. From the options displayed, click **Import**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



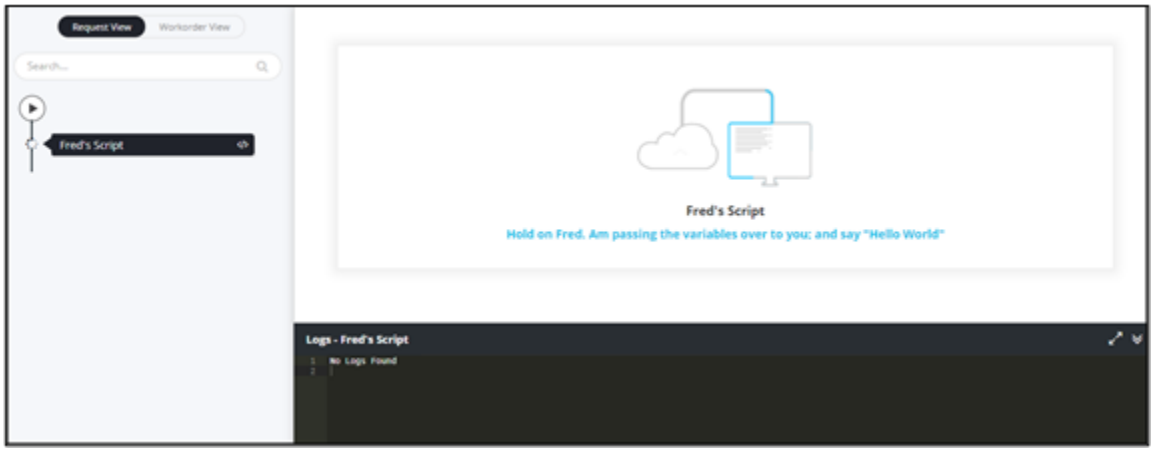
13. In the **Import flows**, window, select the workflow(s) and click **Import**.
14. Drag and drop the imported workflow into the workspace.



15. Refer the Global variables if needed.
16. Connect the workflow tasks and enable the workflow.
17. Trigger the workflow.

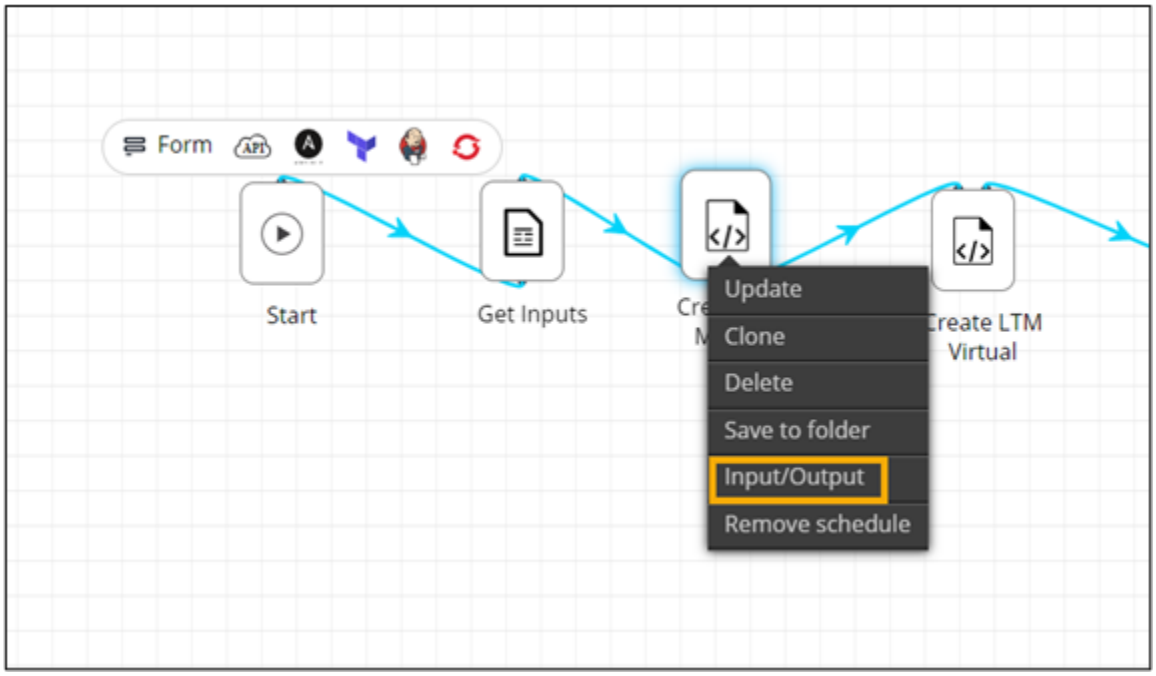
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Workflow executed with the reused task.



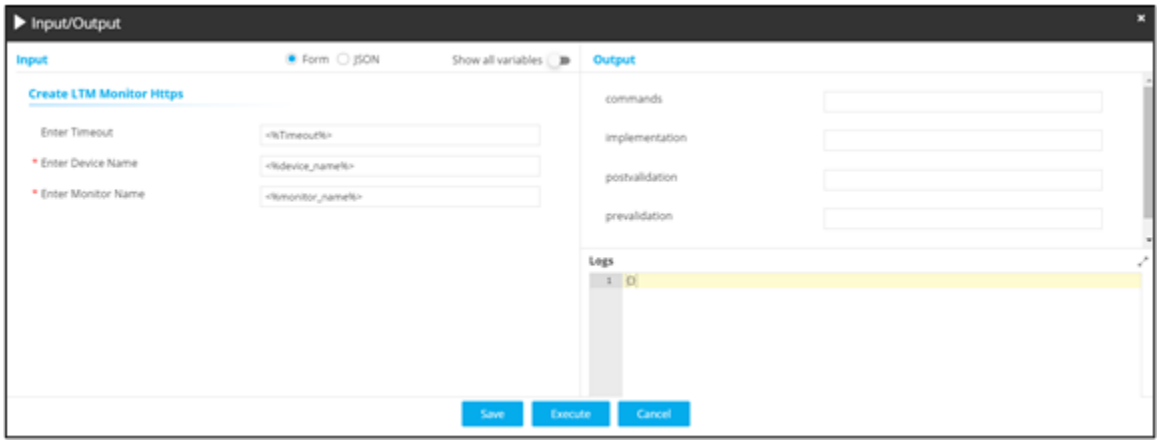
Input/Output Variable Mapping

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.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The input/output variables defined in the task are displayed.



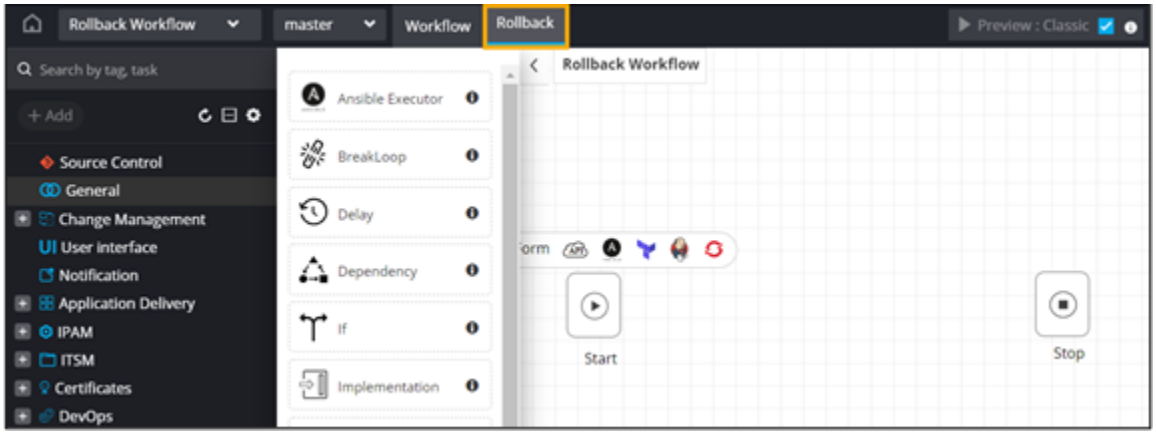
Rollback Workflow

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.

- Manual Rollback
- Auto Rollback

Manual Rollback

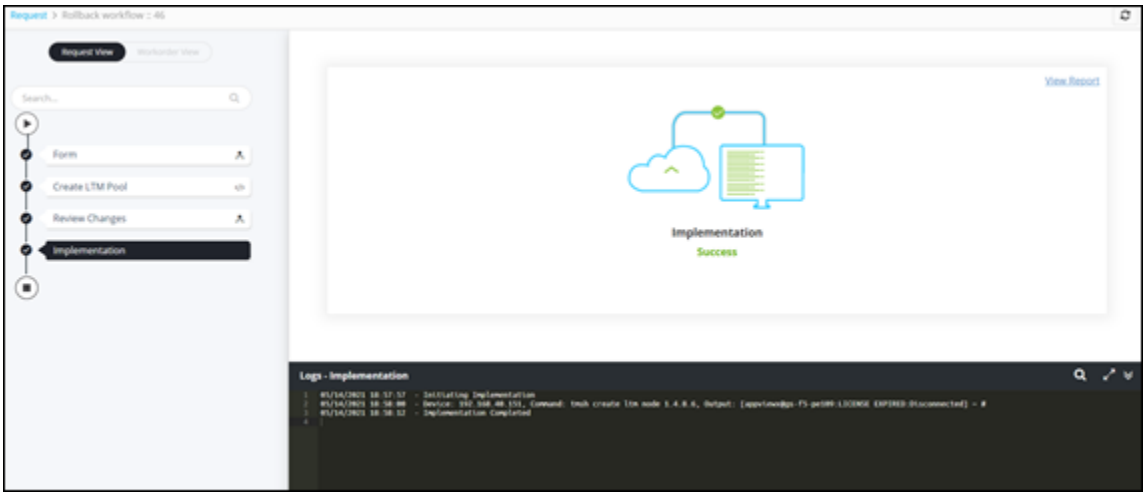
1. Design a workflow using workflow tasks.
2. To add a rollback workflow, from the menu on the top, click **Rollback**.



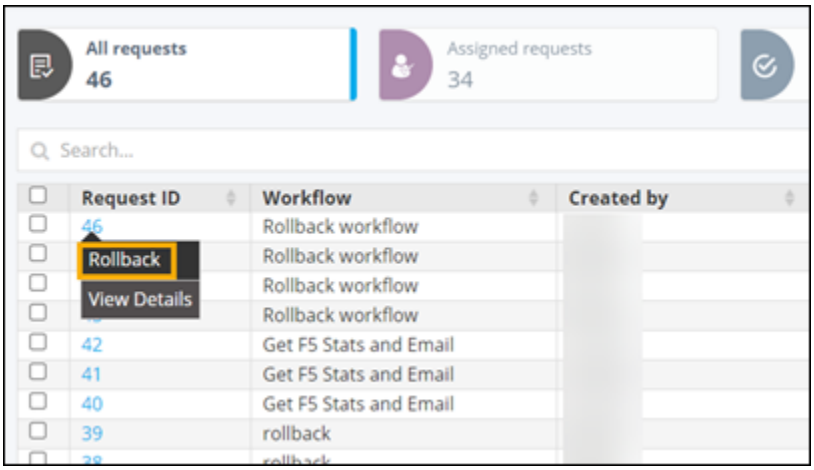
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. From the General section, drag and drop the **Rollback** task.
4. Connect the workflow tasks and enable the workflow.
5. Trigger the workflow.

   Workflow is executed successfully.



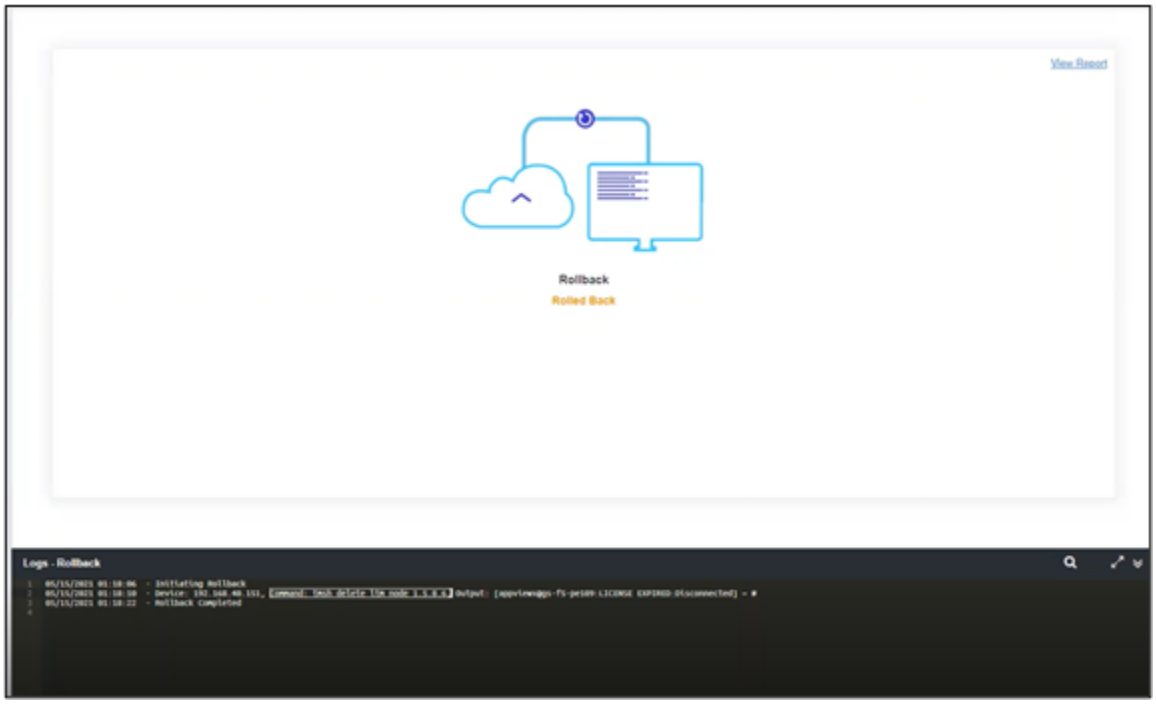
6. On the Request : All page. right-click on the workflow Request ID and select **Rollback**.



7. In the Rollback window, select **Rollback type** as **Request**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

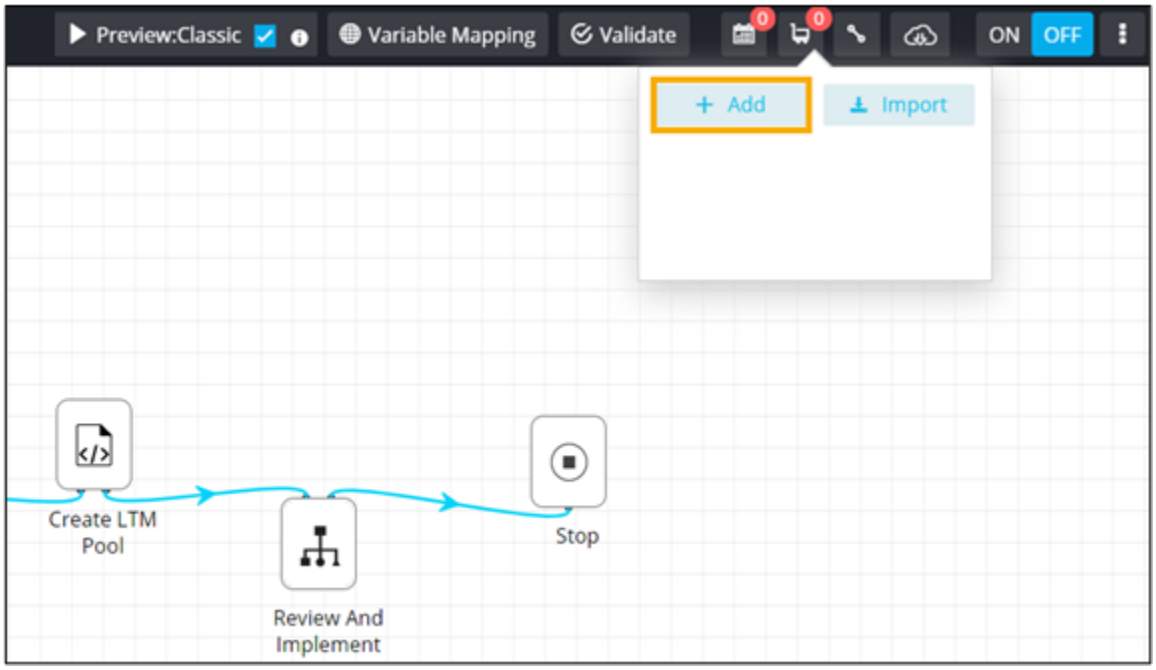
Workflow is **Rolled Back**.



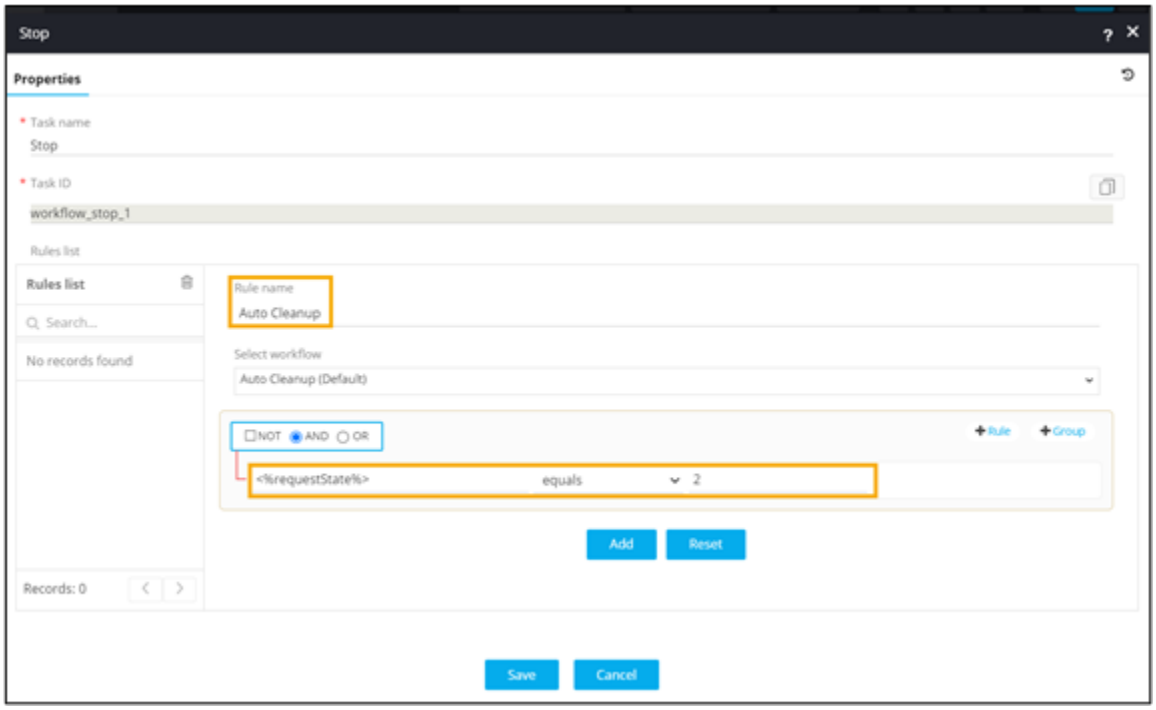
Auto Rollback

1. Design a workflow using workflow tasks.
2. To create a subflow, from the command bar on the top, click **(Cart)** icon.
3. From the options displayed, click **Add**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. In the New subflow window, enter a workflow **Name**.
5. From the **General** section, drag and drop the **Rollback** task.
6. Double-click on the **Stop** task of your main workflow.
7. In the **Stop** task window, define a rule to trigger the auto rollback.

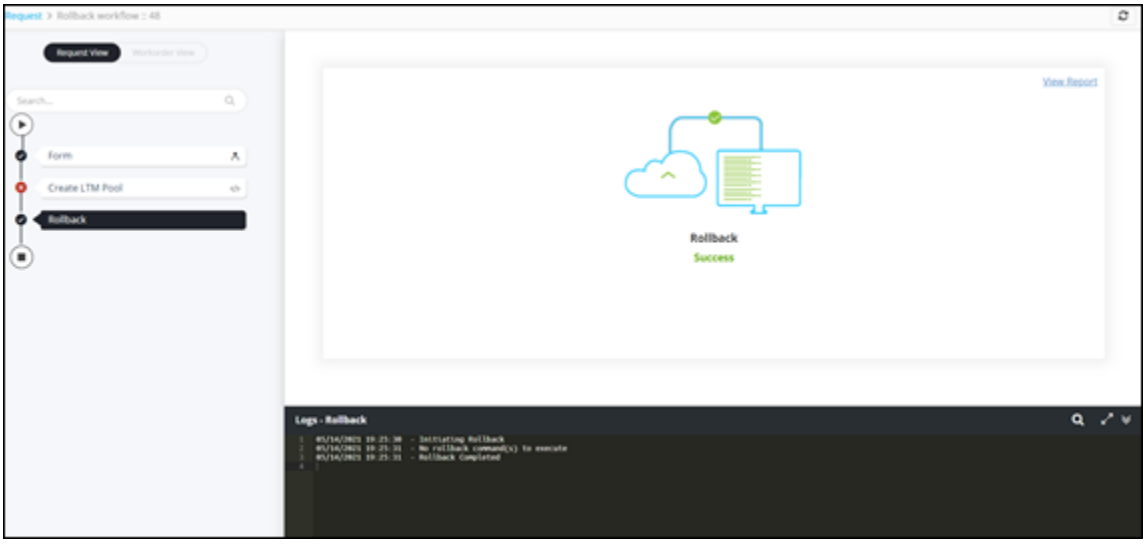


8. Click **Add**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

9. Connect the workflow tasks and enable the workflow.
10. Trigger the workflow.

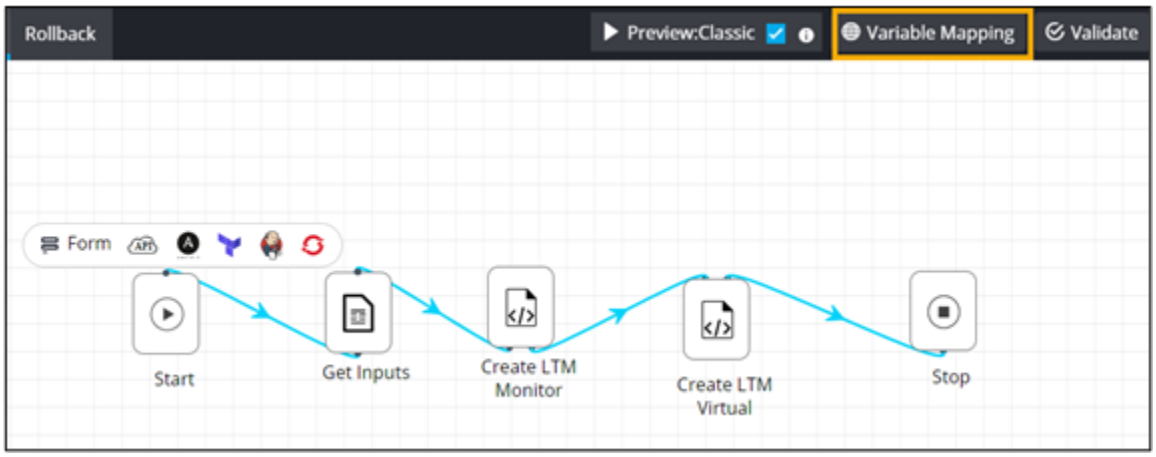
    Auto Rollback is initiated based on workflow state.



Variable Mapping

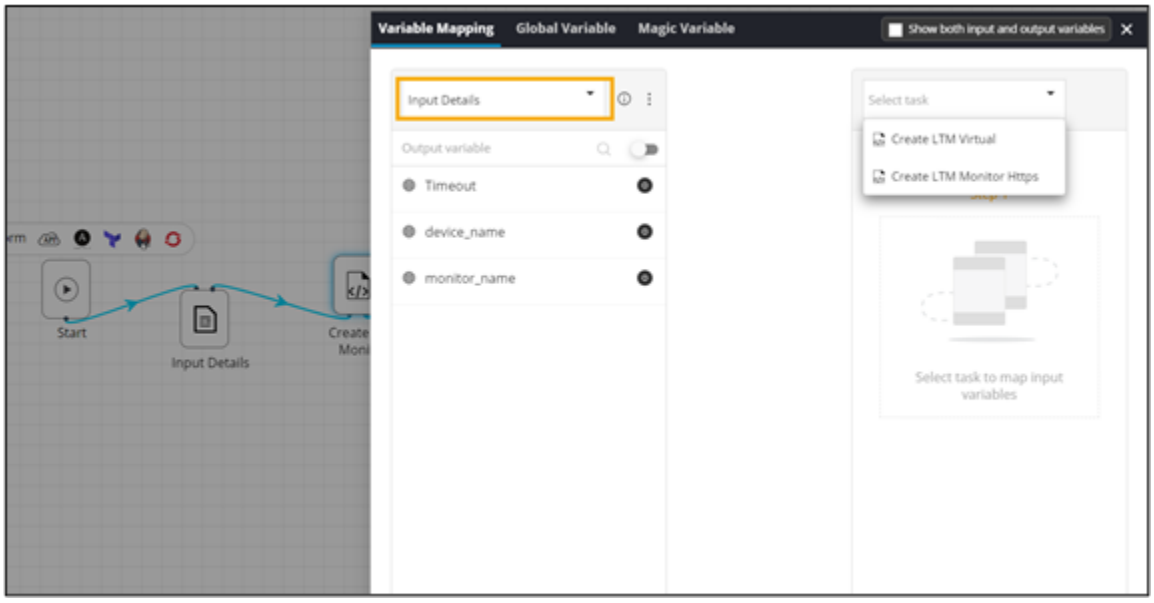
Variable mapping allows you to map the variables between tasks. You can select one or more tasks in order to map variables and get a quick view of the number of variable associations across tasks.

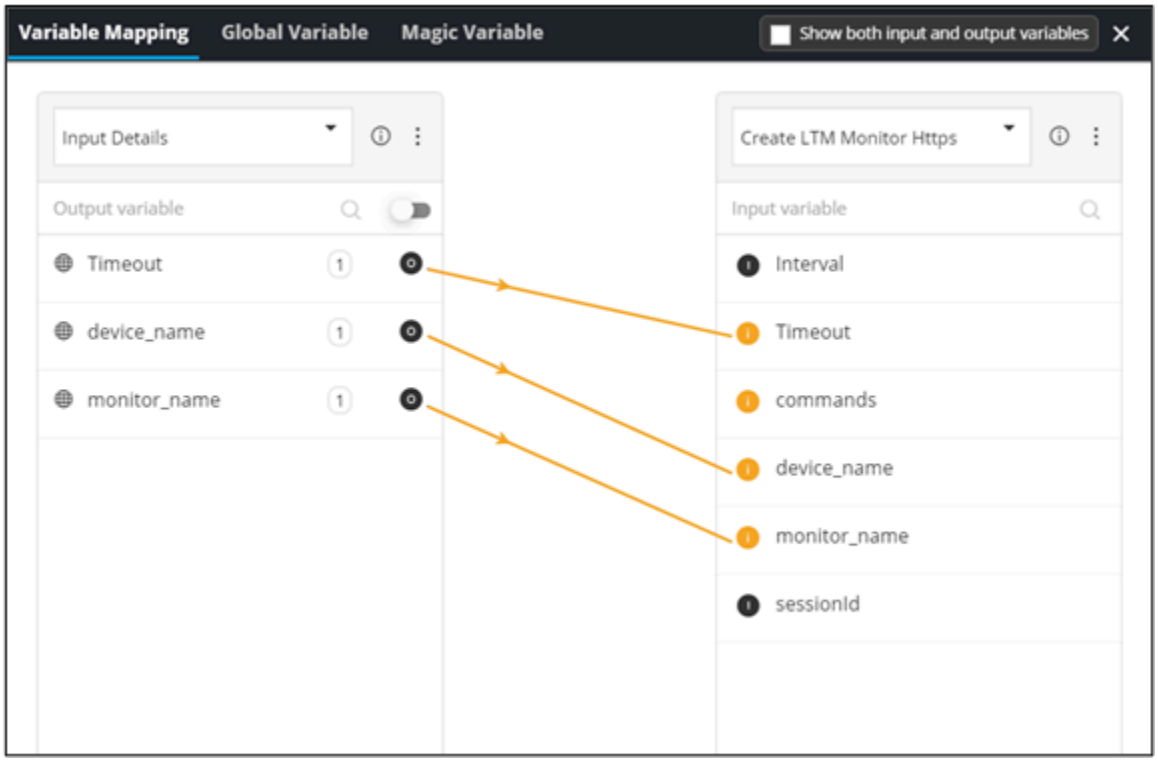
1. Design a workflow or open an existing workflow,
2. From the command bar on the top, click **Variable Mapping**.



3. Under **Variable Mapping**, select the tasks for variable mapping.

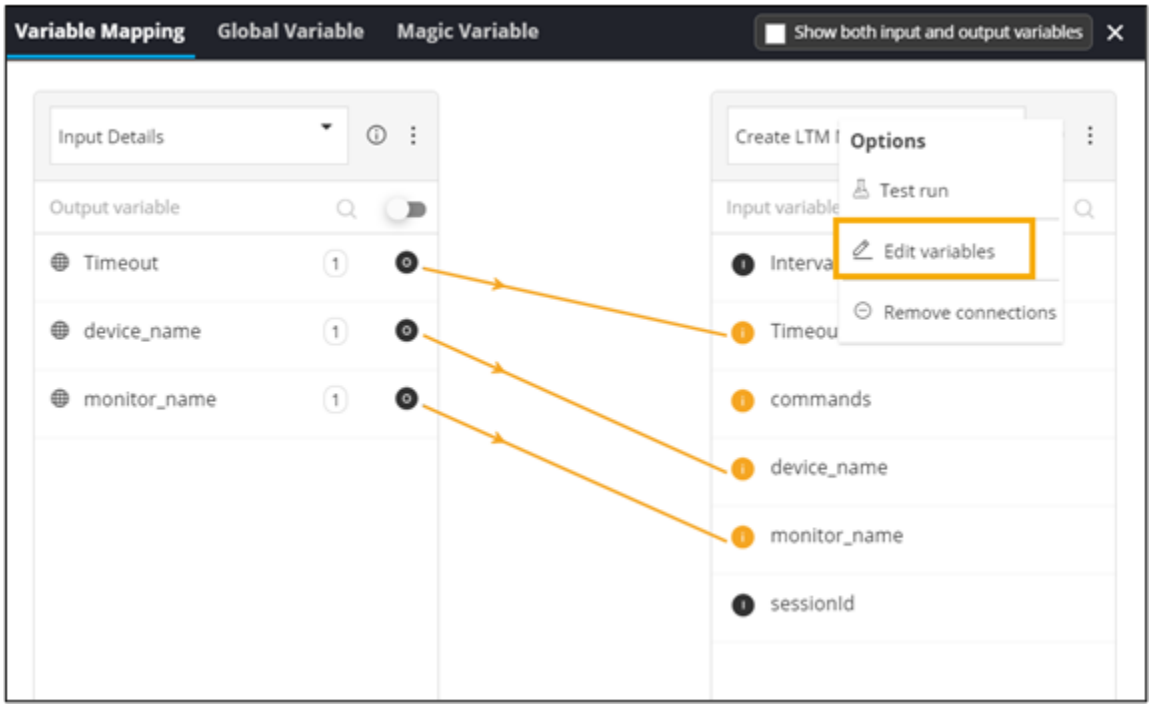
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



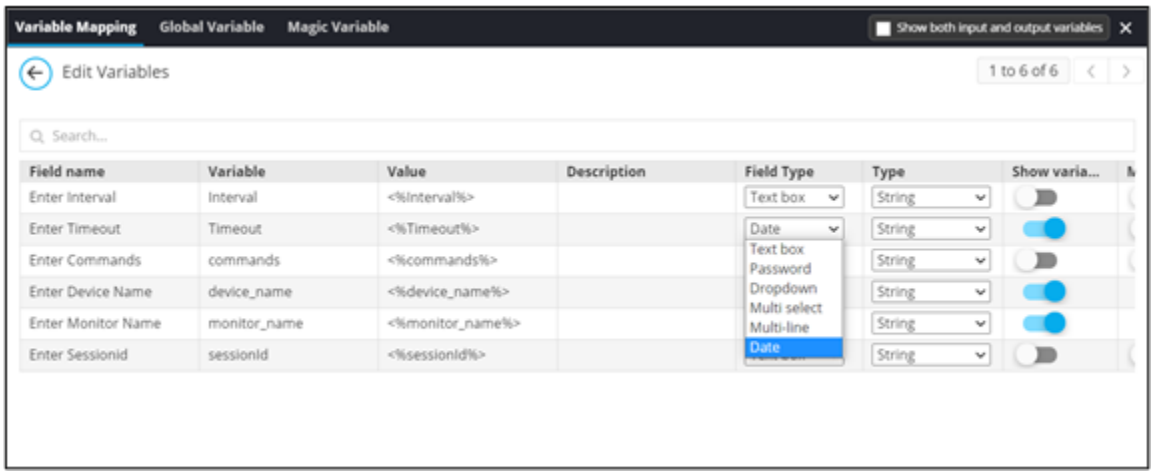


4. To see options, from the top right corner of the window, click ⋮.
5. To make changes in the field parameters, select **Edit variables**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Edit the parameters of the variables as required.



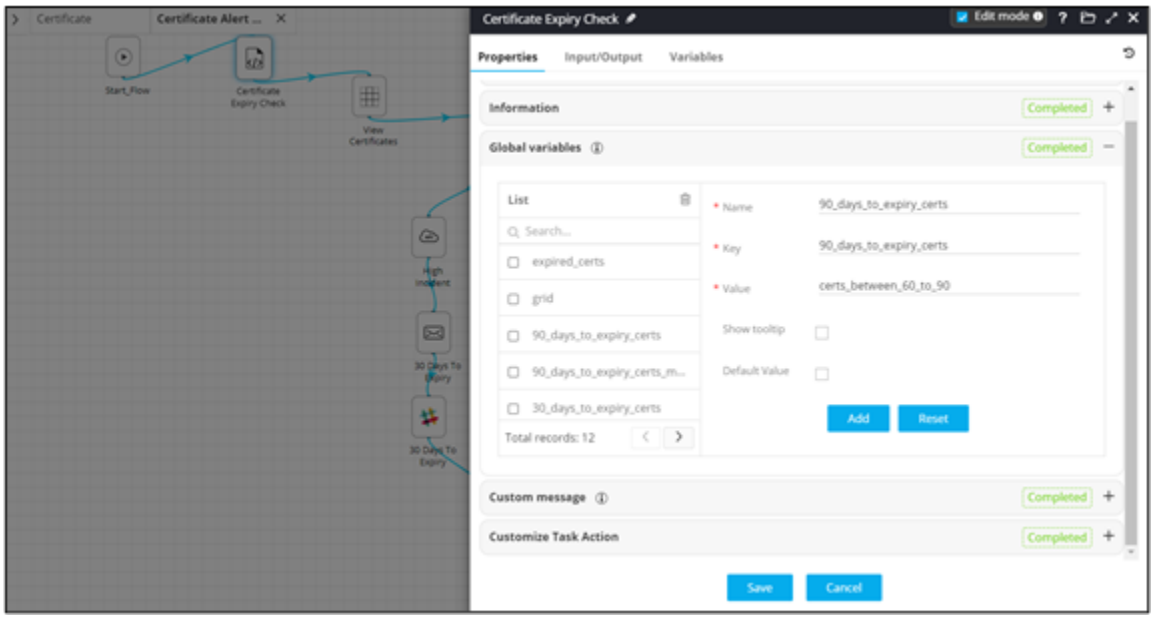
- Global Variables
- Magic Variable

Global Variables

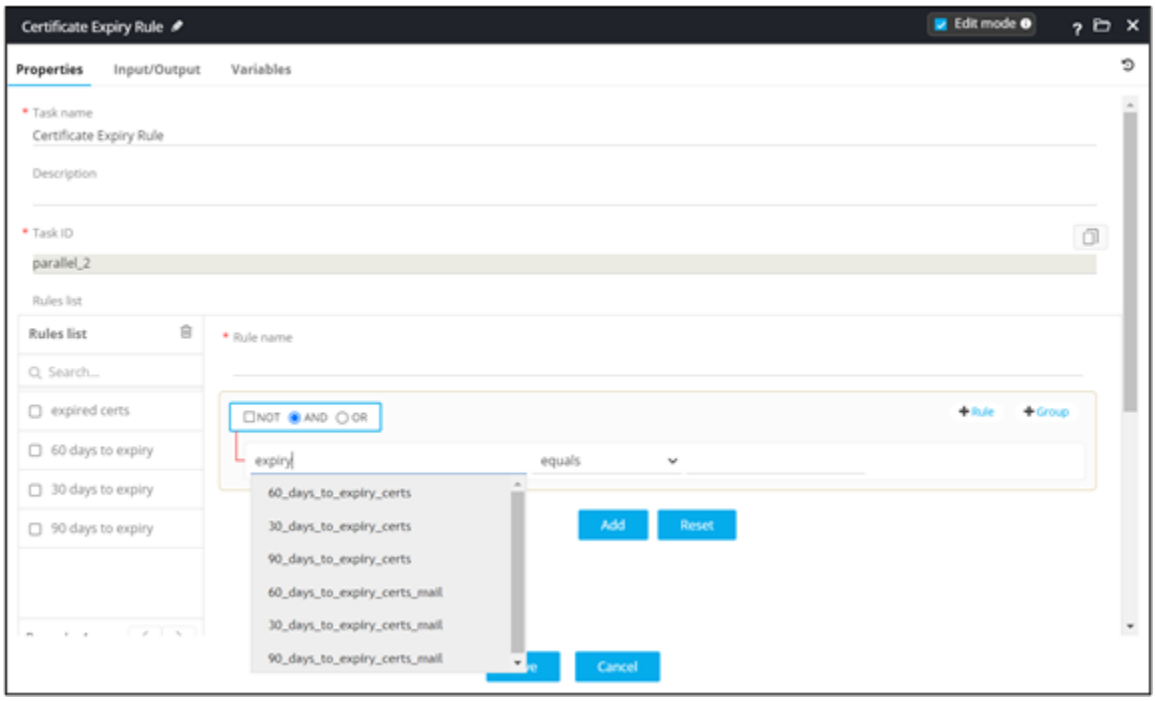
Variables can be declared globally and referenced across any stage of the workflow thus allowing data to flow seamlessly across the entire workflow process.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Design a workflow.
2. Click on any workflow task.
3. In the task window, under **Properties**, in the **Global Variables** tab, define the **Key** and **Value**.



4. Click **Add**.
5. Auto-populate and refer the global variables using keyword match.



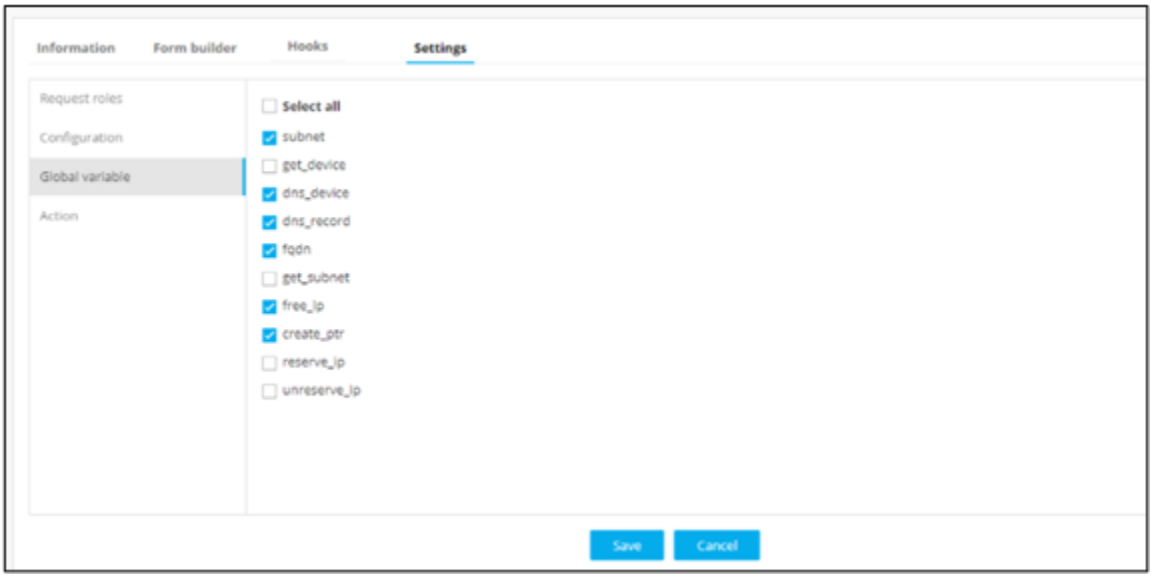
- Declaring Variables from Tasks
- Referring Variables

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

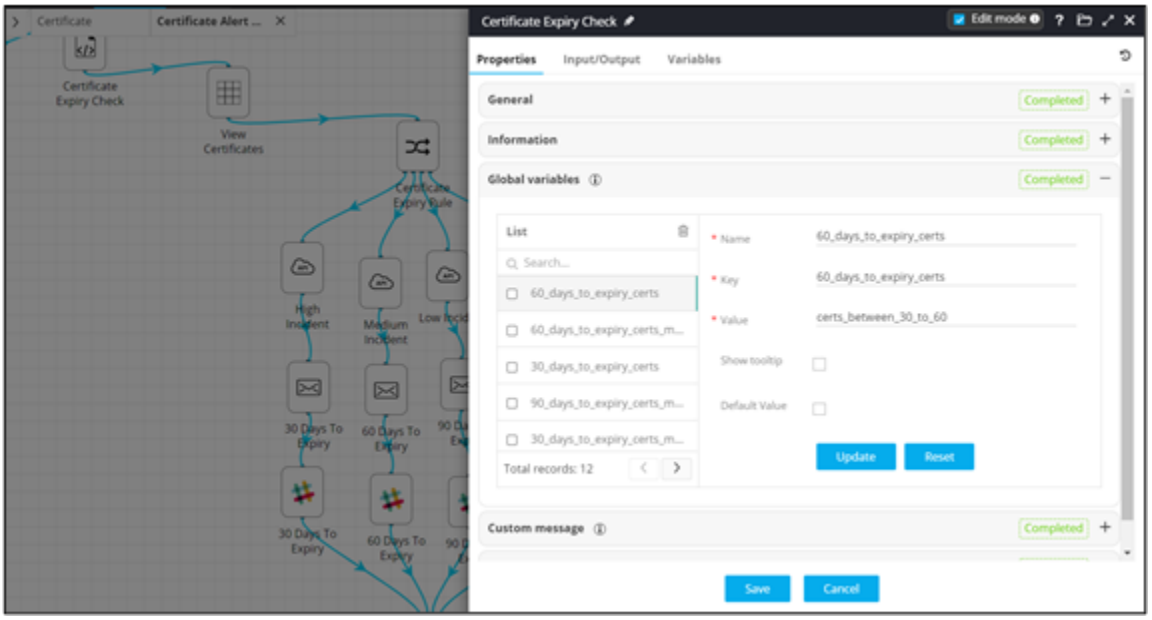
- Global Variable Inventory
- Configuring Tooltips

Declaring Variables from Tasks

- Declaring variables from a Form task

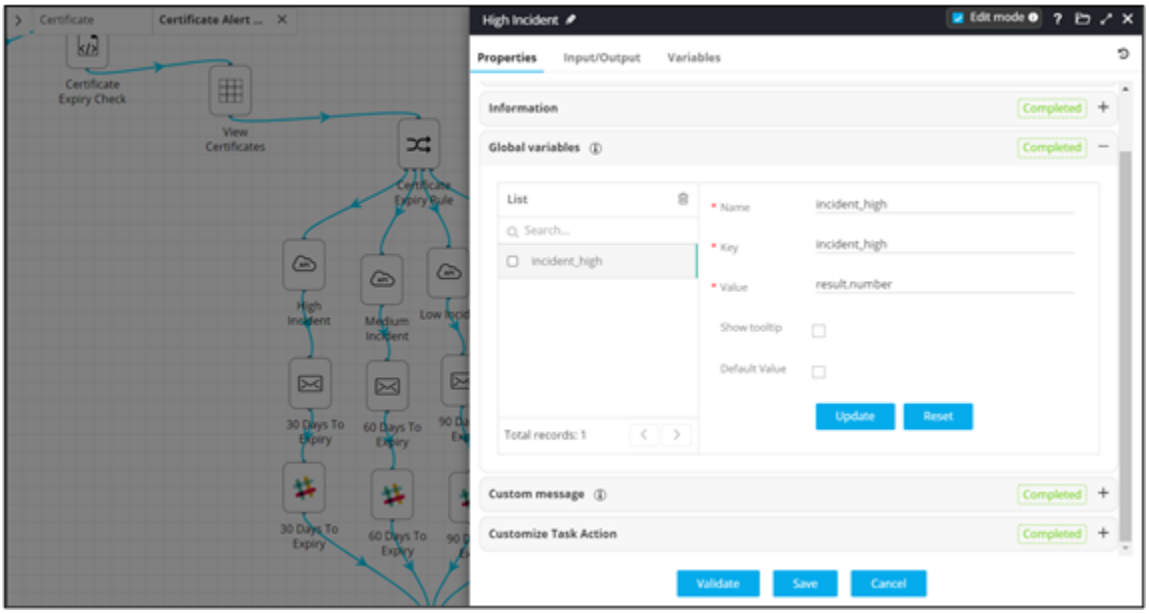


- Declaring variables from a Script task



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• Declaring variables from a REST API task.



Note: Form fields that are declared as global variables in the request form will now be available as query parameters in the URL available in the request stage view. On clicking the URL with query params framed using the field id of the form as keys and global variable values as values, you will be redirected to the request creation page with pre-filled forms with values from the query params of the URL.

Referring Variables

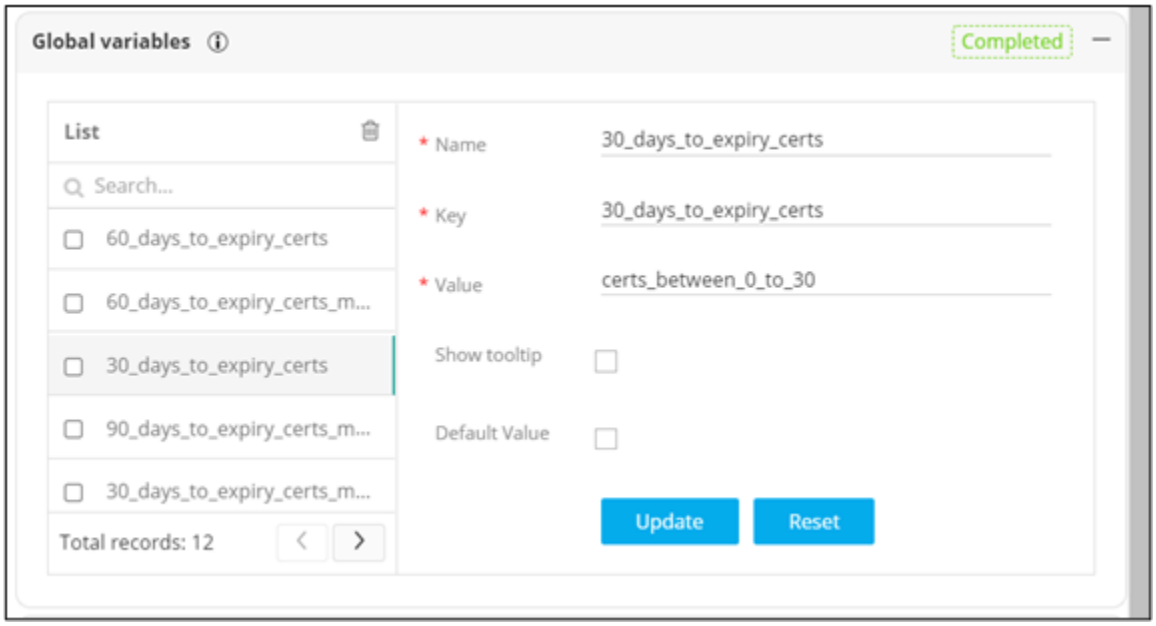
Once declared, global variables can be simply referred across any other stage using the following syntax:

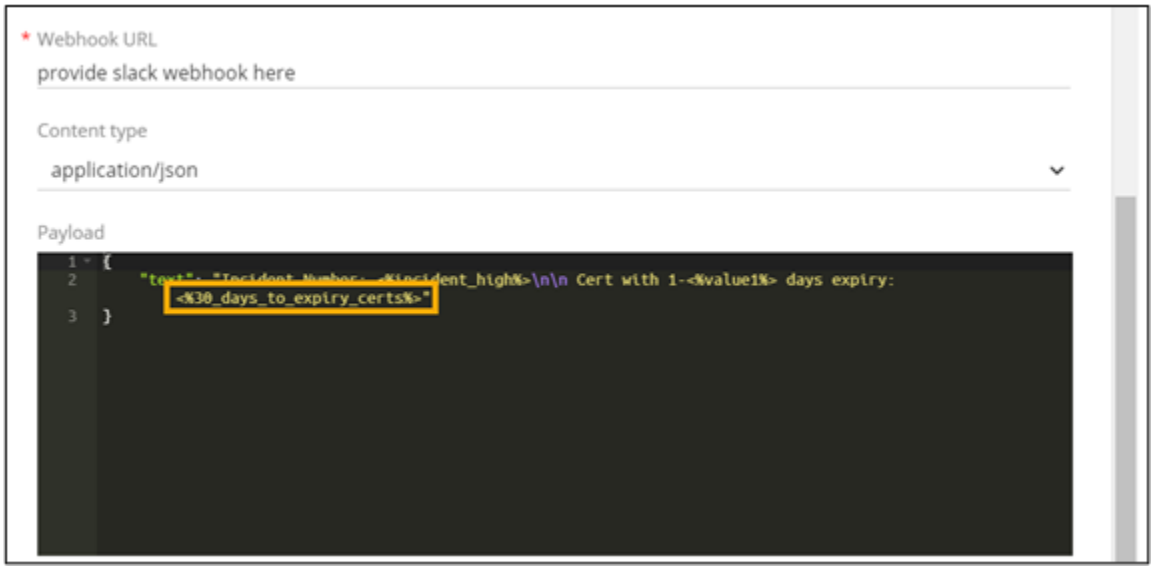
`<%variable%>`

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%>`

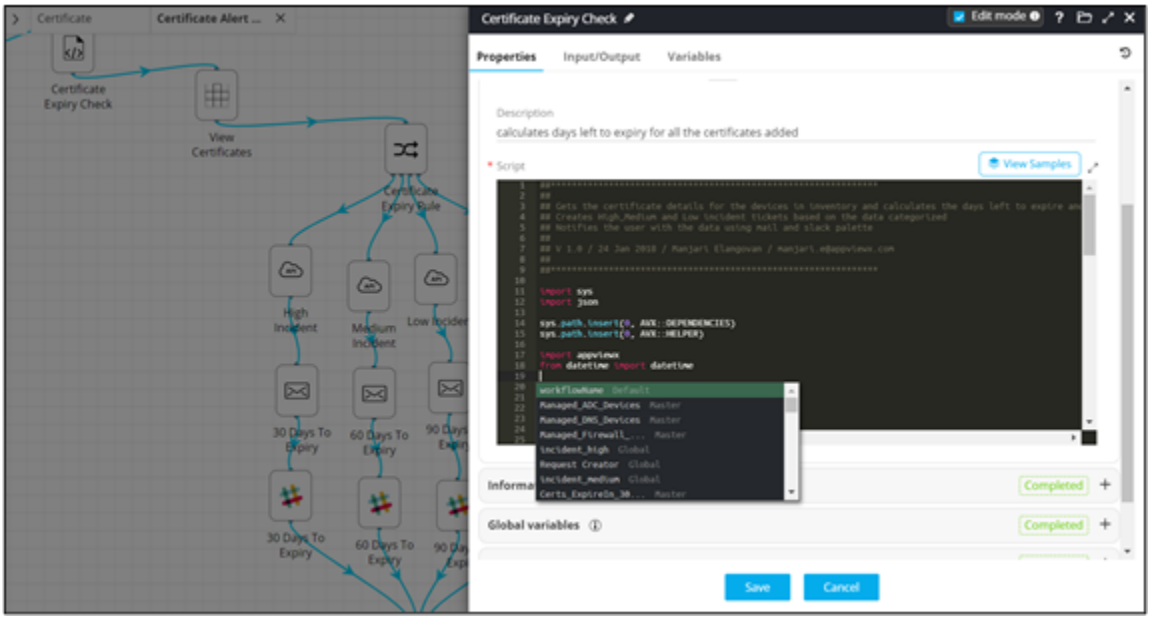
Copyright © 2025 AppViewX, Inc. All Rights Reserved.





Note: Use keyboard shortcut **Ctrl+g** to view and refer global variables within a Script task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

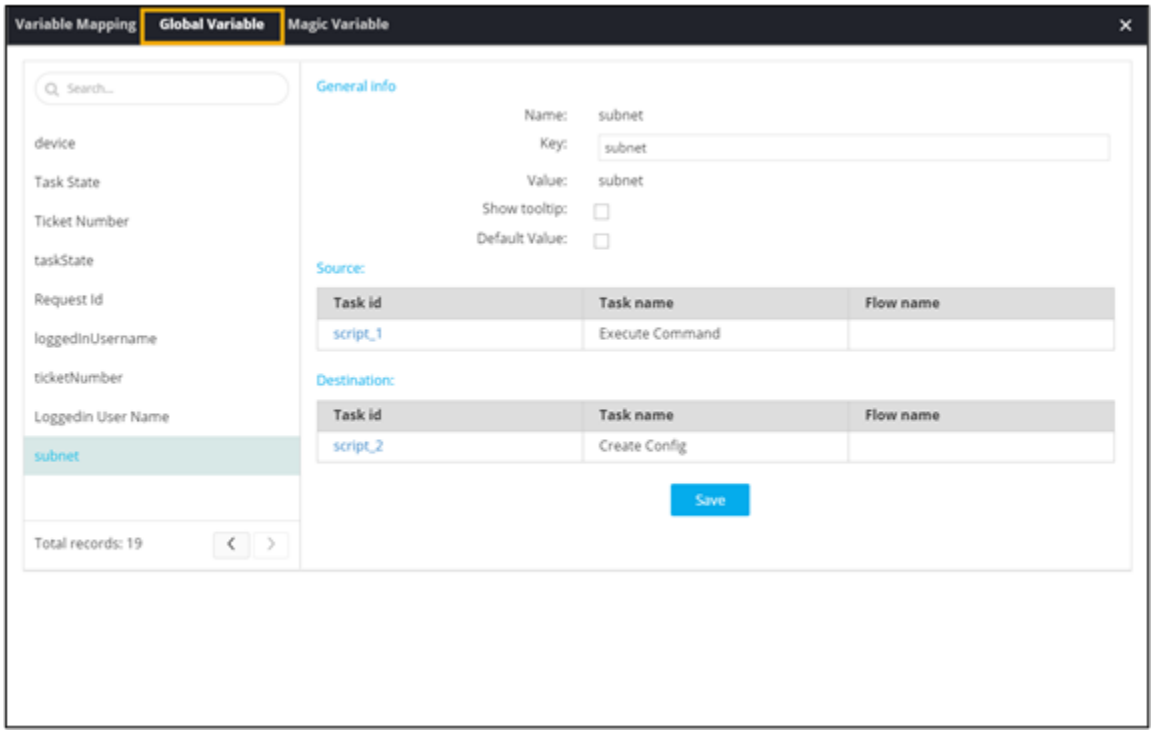


Global Variable Inventory

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 © 2025 AppViewX, Inc. All Rights Reserved.



Note: Any updates to the Key field will be reflected globally across the workflow.

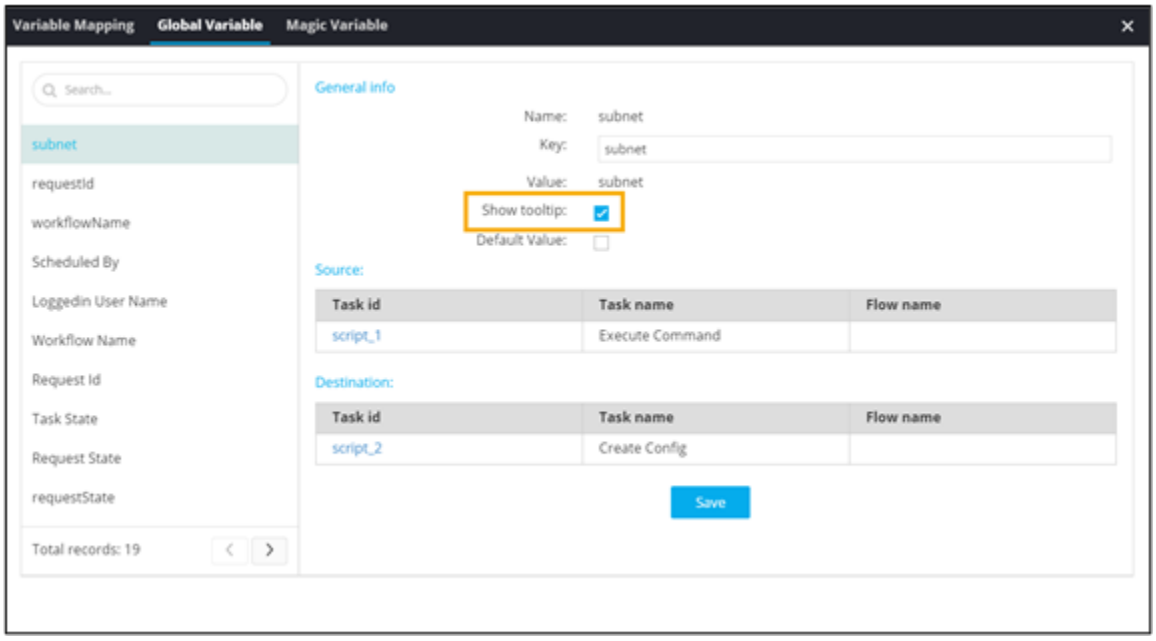
Configuring Tooltips

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:

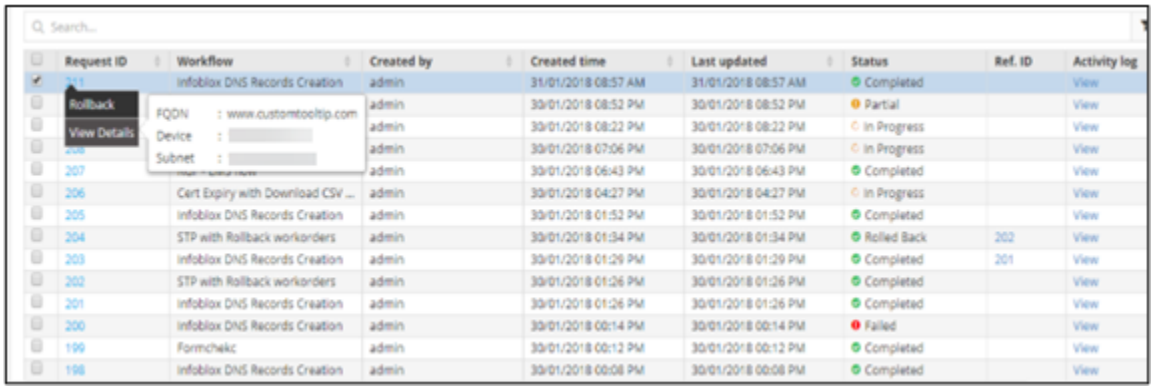
1. Design a workflow.
2. Drag and drop the required workflow tasks.
3. Assign the required values as global variables.
4. To view all global variables, from the command bar on the top, click **Variable Mapping**.
5. Under **Global Variable**, select the **Show tooltip** checkbox.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Connect the workflow tasks and enable the workflow.
7. Trigger the workflow.

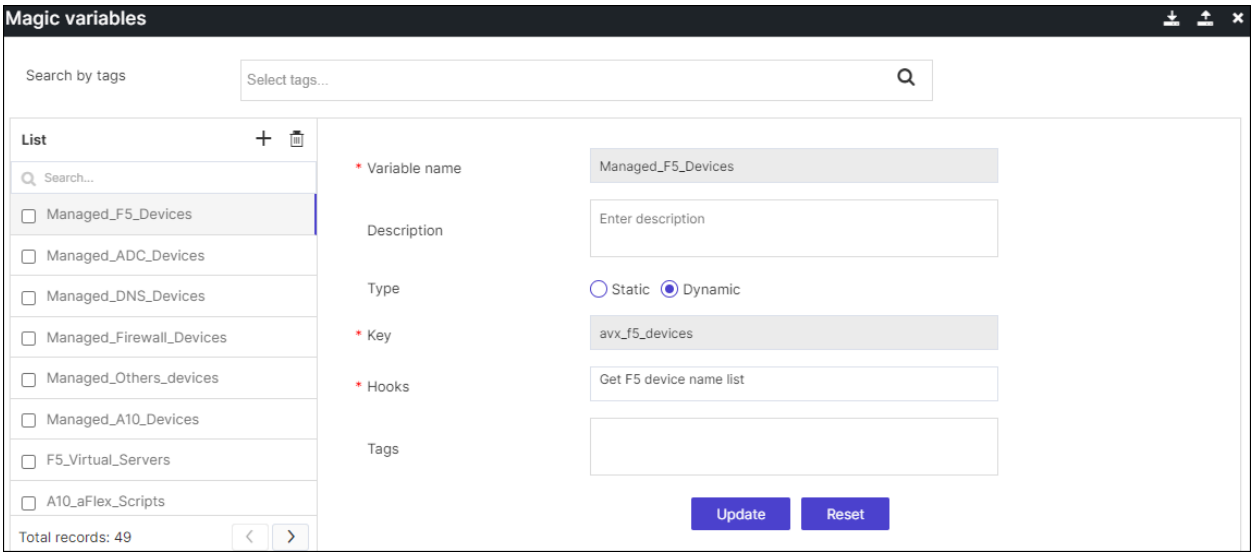
   Workflow is executed successfully.
8. On the Request :: All page, right-click on the Request ID of the workflow.
9. To see the tooltip information, click **View Details**.



Magic Variable

This shows a list of all the Magic variables defined in various tasks across workflows.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



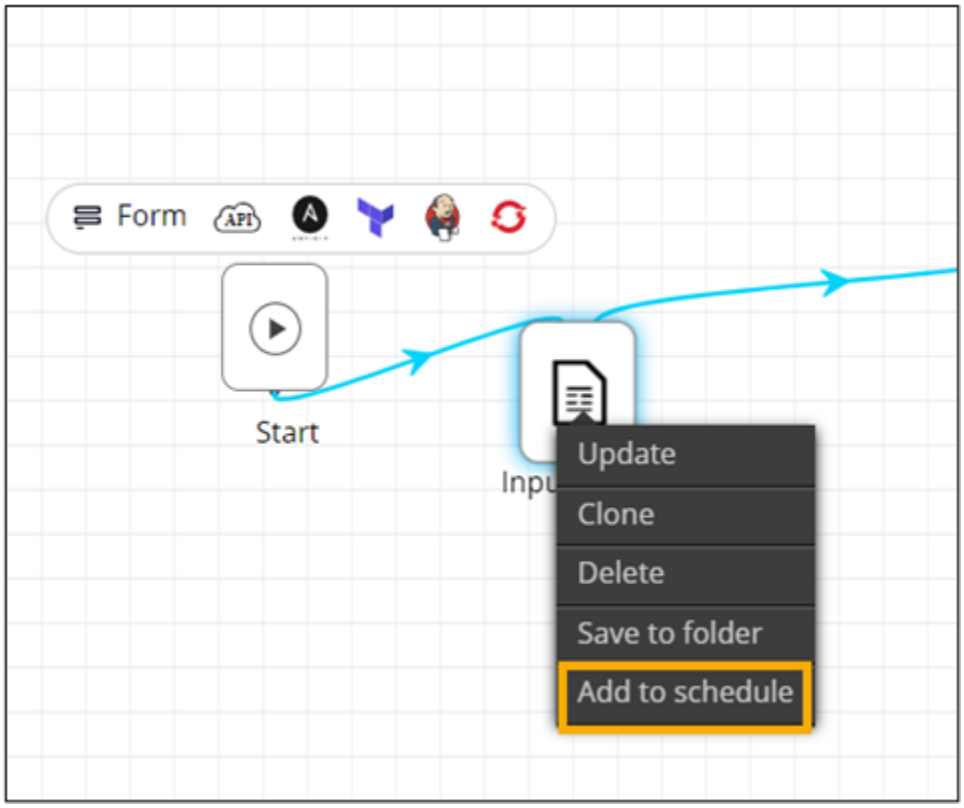
- Accessing Magic Variables
- Defining a Hook and declaring it as a Magic Variable
- Using a Magic Variable within a Workflow

Scheduling Tasks

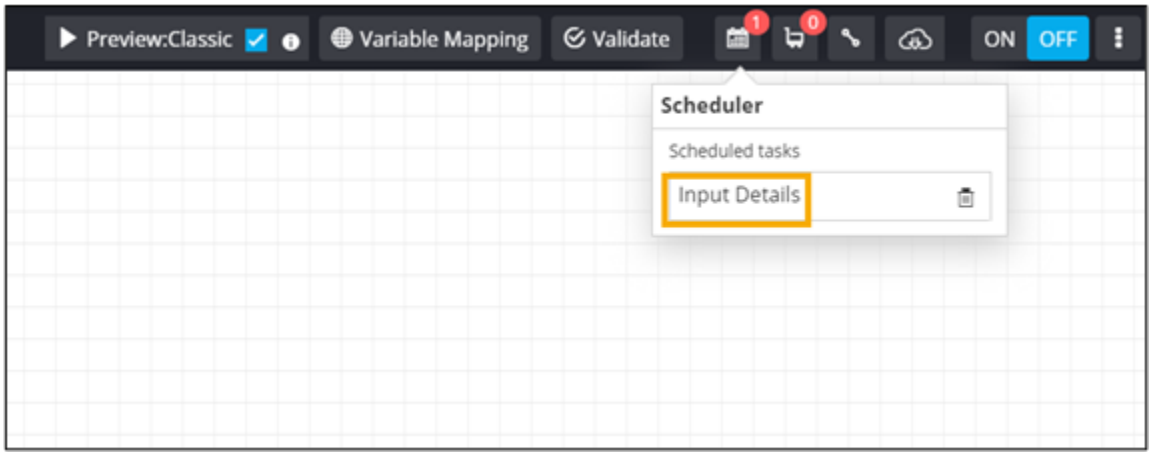
You can schedule workflows and also schedule specific tasks within the workflows that require user inputs.

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**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. Click **Yes** in the **Confirmation** pop-up window.
   The task is added to the **Scheduler**.



5. Connect the workflow tasks and enable the workflow.
6. On the Request :: View/Run page, search for the workflow and schedule the workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The request is added to the **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.

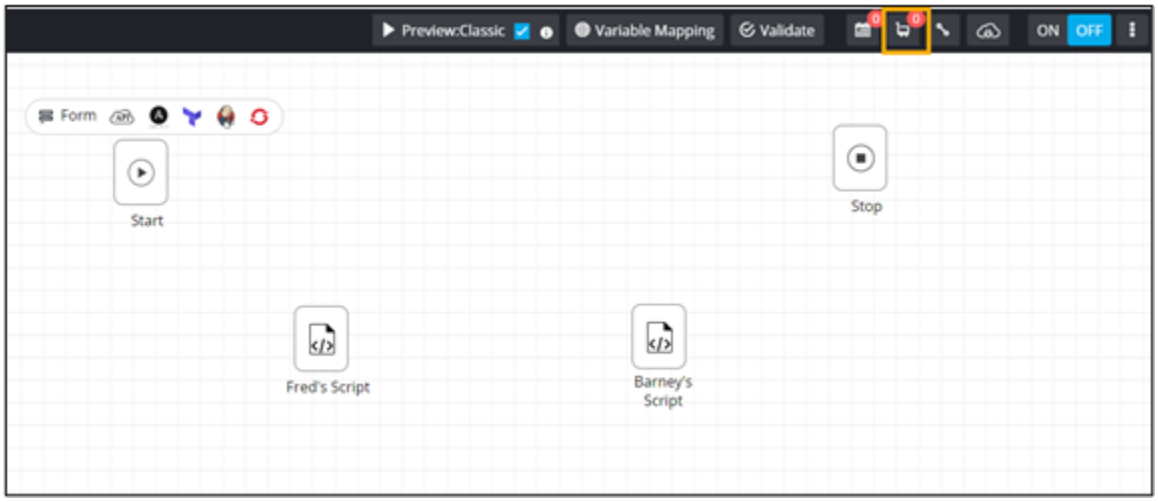
Note: Relevant global variables must be declared and referenced in order for nested workflows to work seamlessly.

• How to Add/Import a Nested Workflow?

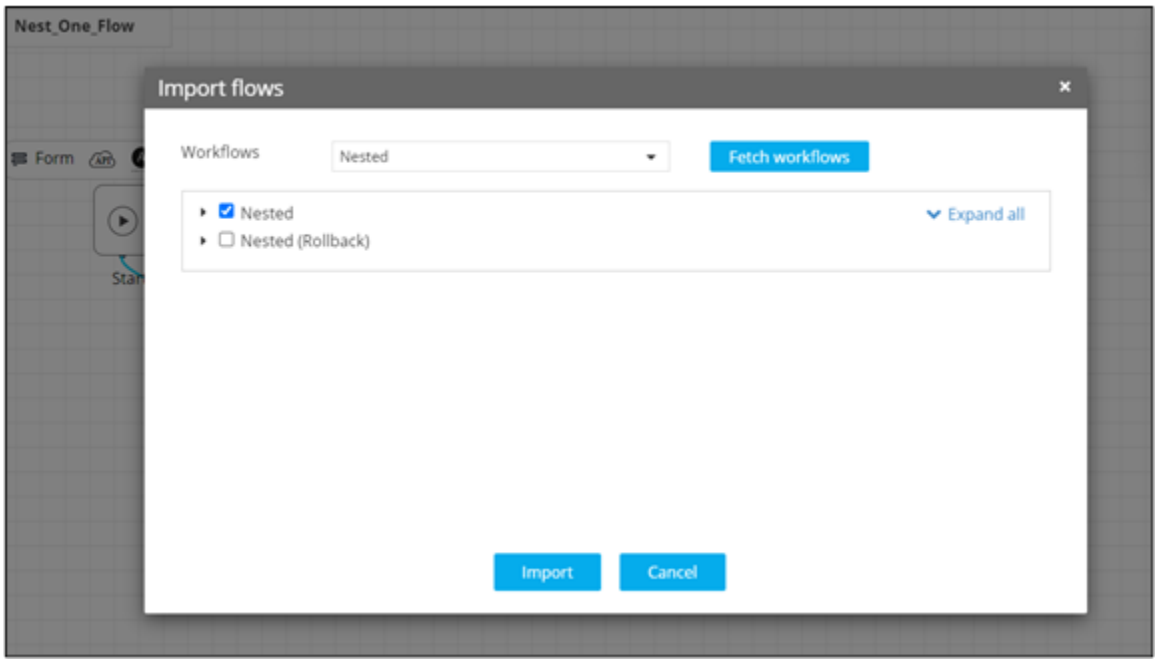
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

How to Add/Import a Nested Workflow?

1. Design a new workflow.
2. To add or import a nested workflow, click **(Cart)** icon.



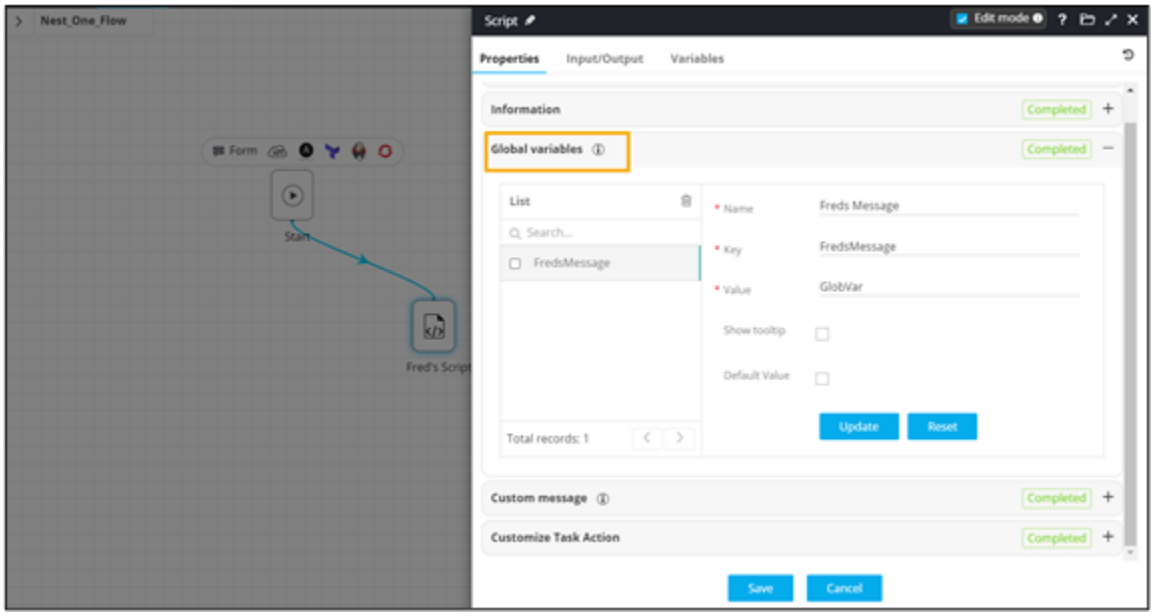
3. To define a new subflow, from the options displayed in the dropdown, click **Add** or to reuse an existing workflow, click **Import**.
4. In the **Import flows** window, select the workflow to be reused or nested.



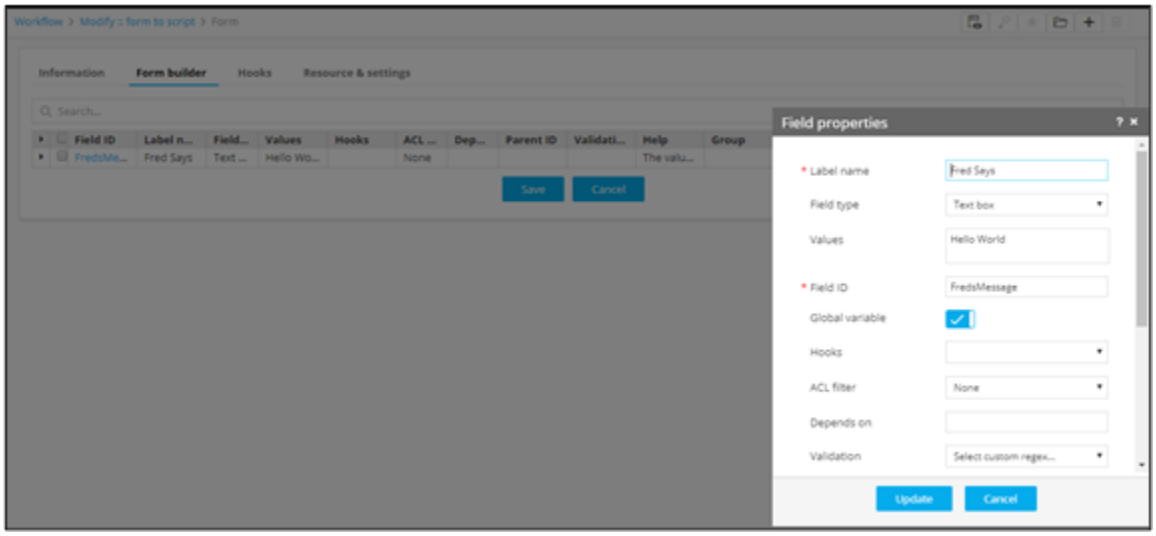
5. Drag and drop the workflow to be nested into the workspace.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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.



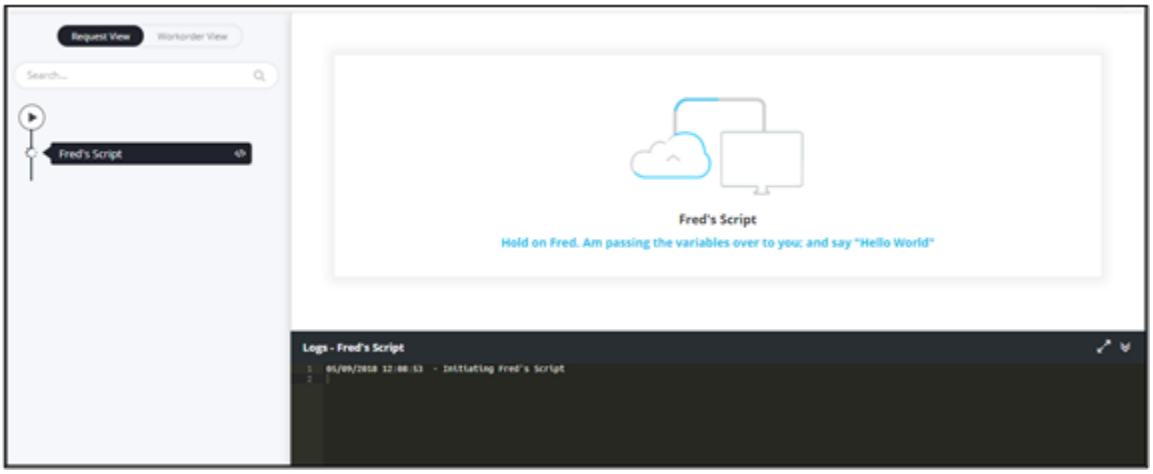
7. Refer the Global Variable key into one or more task(s) of the nested flow.



8. Connect the workflow tasks and enable the workflow.
9. Trigger the workflow.

Copyright © 2025 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 referred.

Version Control

You can design and have multiple versions of a workflow and enable one master version.

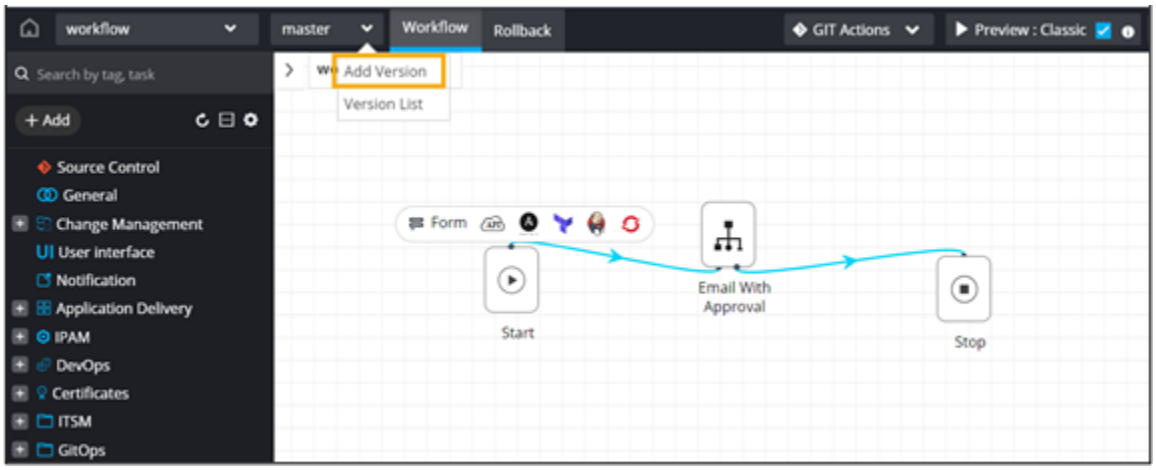
- Provision to version control the workflows.
- Provision to upload and audit requirements/artifacts against each version.
- Version control allows for tracking up to ten versions of the workflow.
- Provision to update, modify version.
- Provision to audit changes made to workflow tasks via ‘Task history’.

- Adding a Version
- Viewing the Workflow Version List

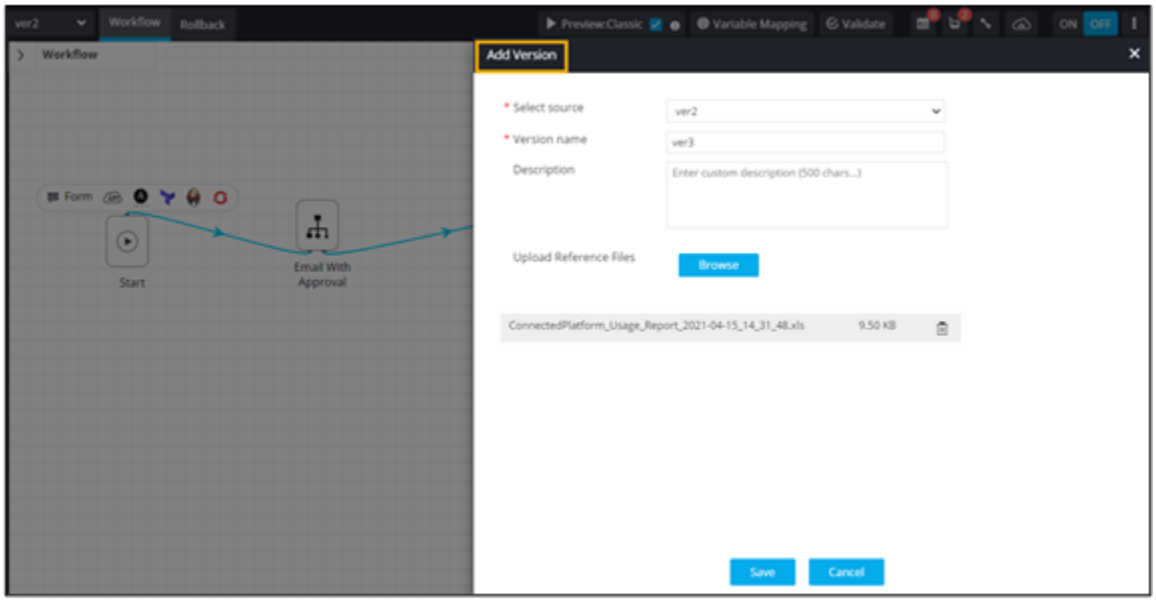
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Adding a Version

1. Design a workflow.
2. From the master dropdown menu, select **Add Version**.

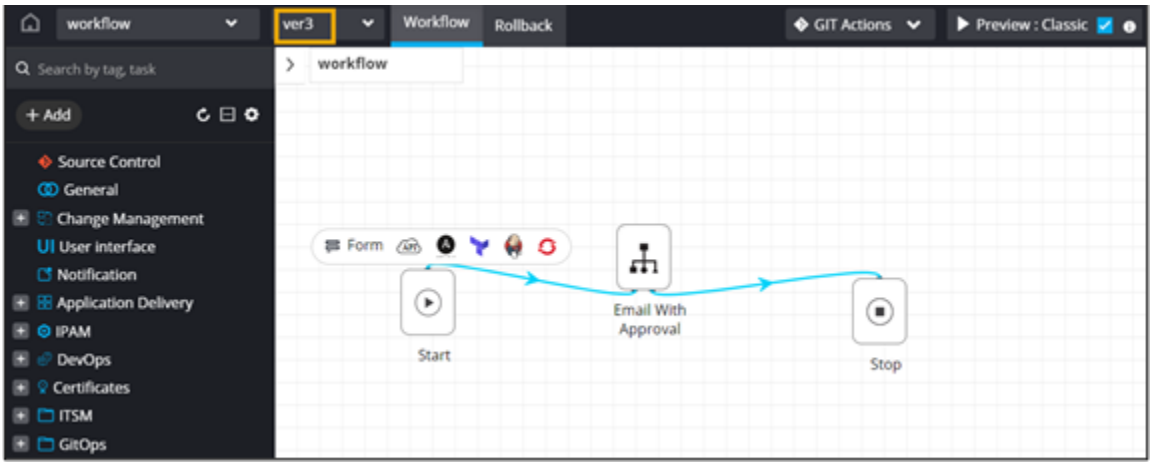


3. In the **Add Version** window, add the version details and click **Save**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

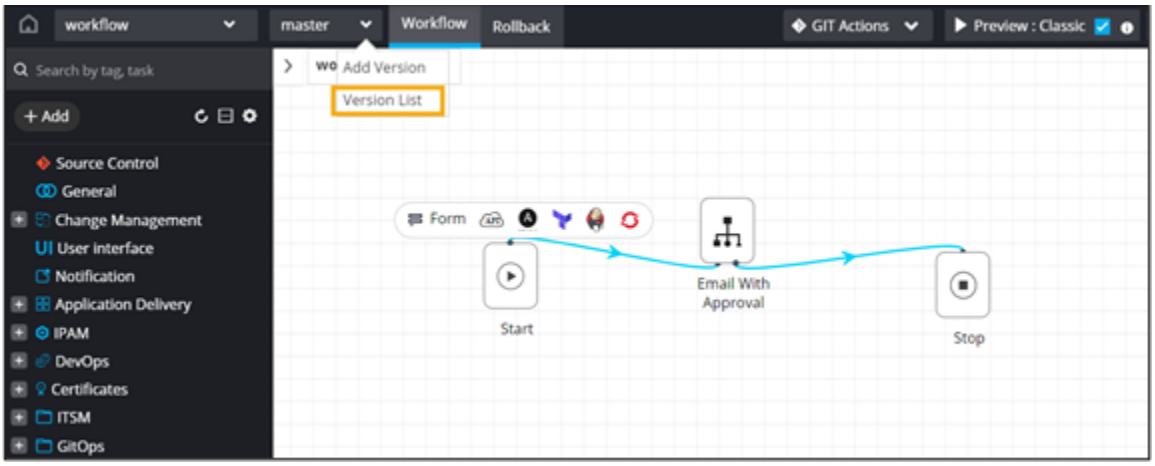
- The newly added workflow version is activated and the workflow version is also displayed on the Workflow page.



Viewing the Workflow Version List

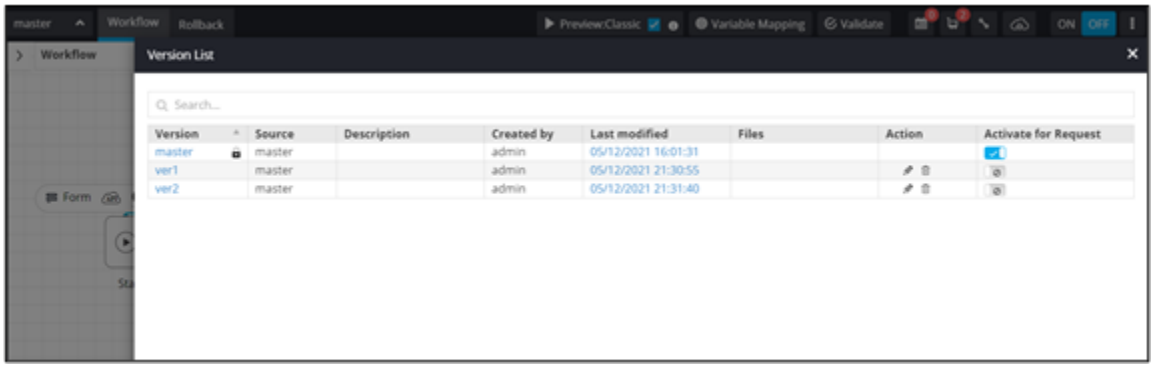
To check the number of versions created for a workflow:

1. Go to the Workflow page.
2. Open the workflow for which you want to check versions.
3. From the master dropdown menu, select **Version List**.



The **Version List** window displays the list of all versions existing for the workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. To enable a particular workflow version, turn on the toggle.

> **Note:** Only one version can be enabled at a time.

The enabled version is added to the **Workflow** page.

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).

- Preview
- Workflow Alignment
- History
- Workflow Settings
- Validating a Workflow
- Switching between Edit mode/Citizen mode

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

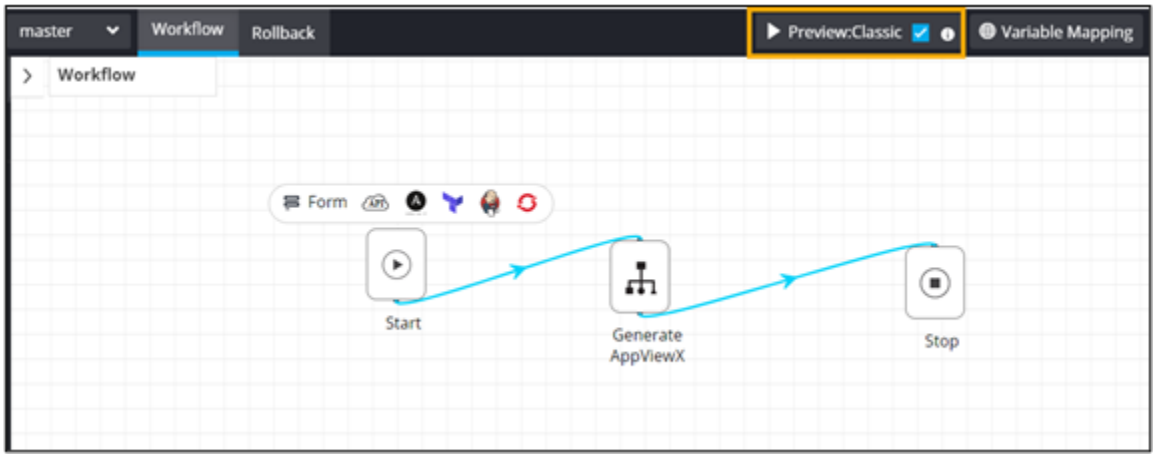
Preview

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
  - REST (I)
  - Schedule
  - Email
  - Slack.

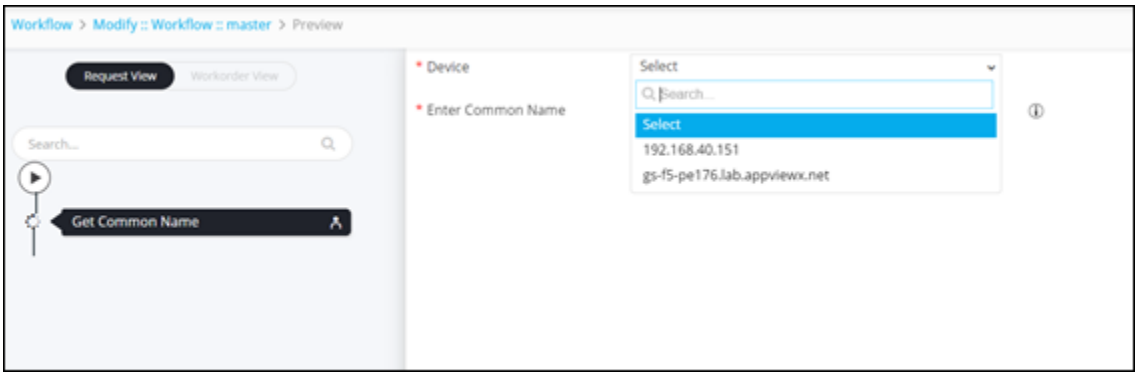
To preview a workflow in Classic mode:

1. Go to the Workflow page.
2. Open a workflow.
3. From the command bar on the top,, select the **Preview:Classic** checkbox.

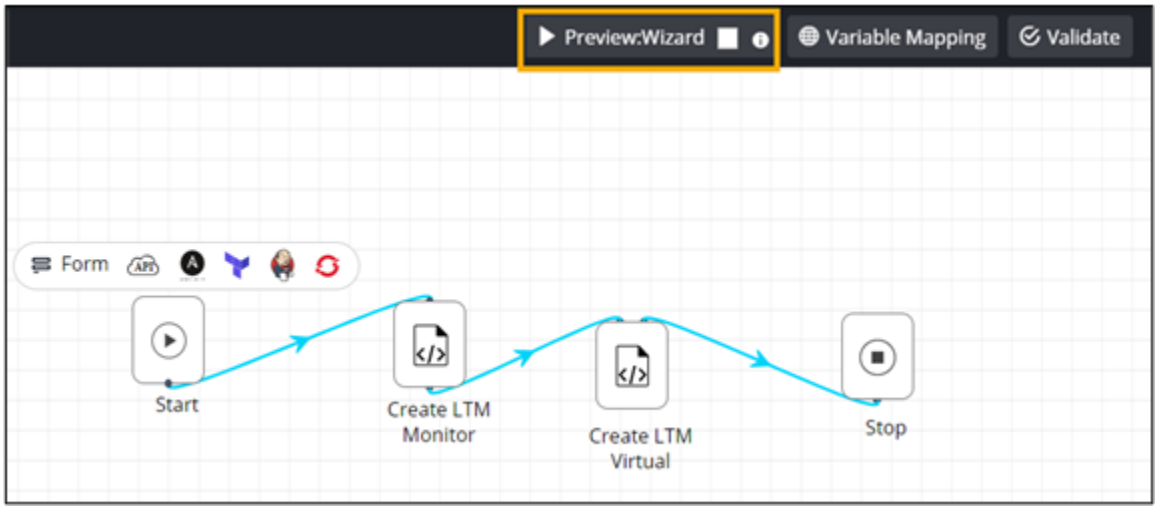


4. Click **Preview**.
   Workflow preview is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



5. To perform Preview in wizard mode, clear the **Preview:Wizard** checkbox.

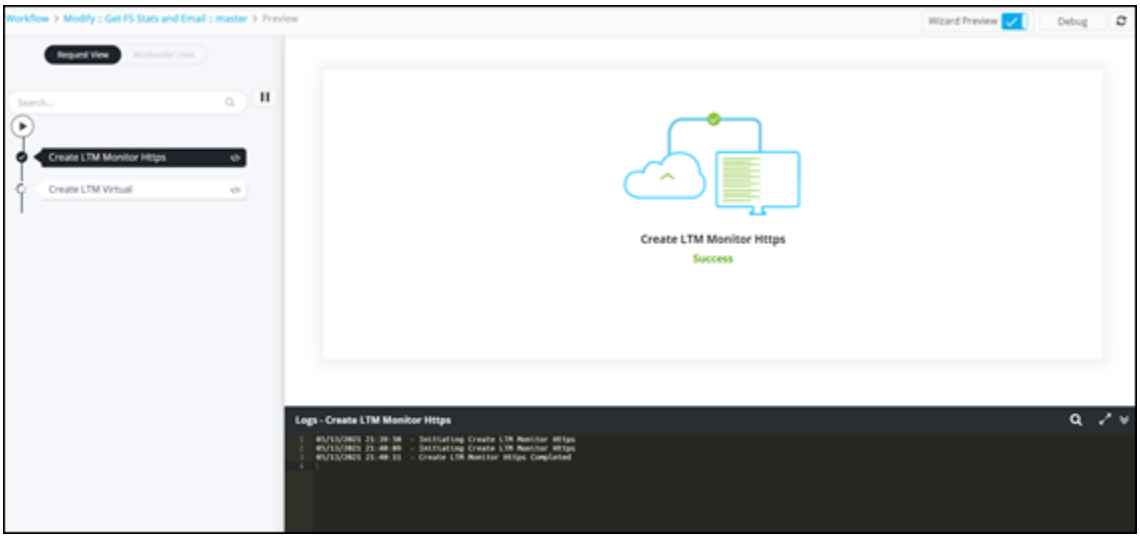


6. Click **Preview**.

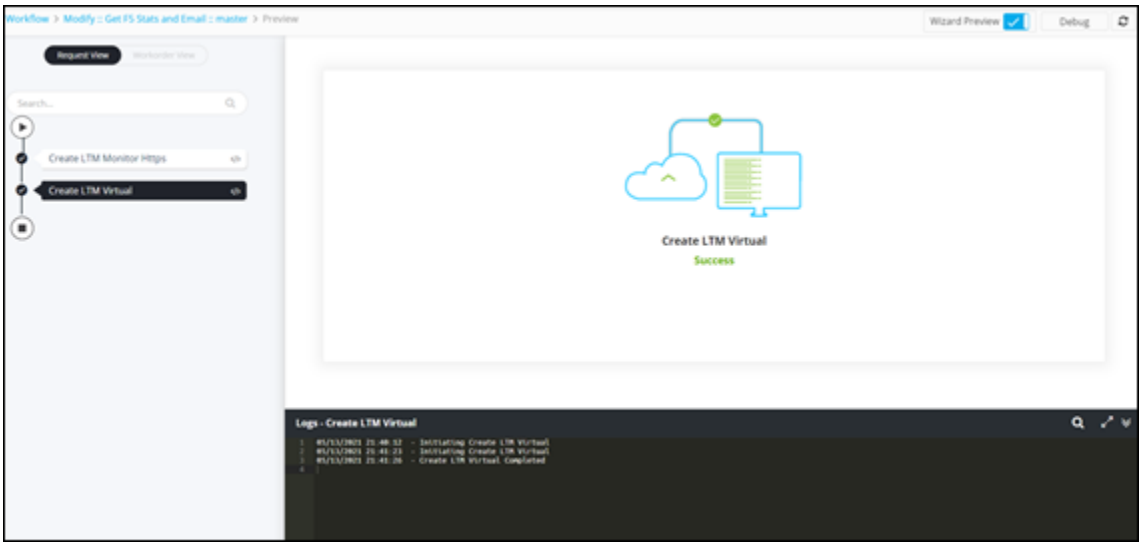
   A pseudo form is generated with the input variables in the Script task.

7. Enter the field information to execute the task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The second task executes automatically once the first task is successful.

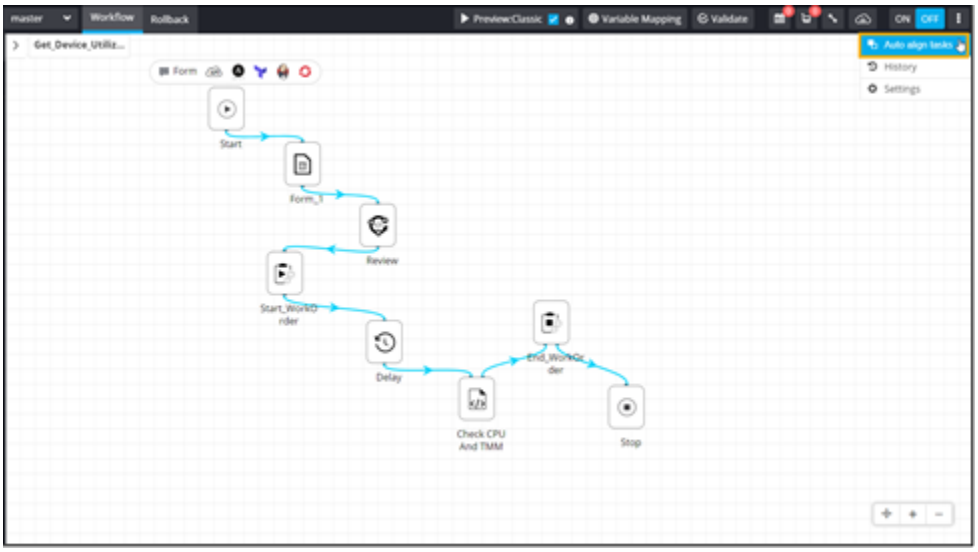


Workflow Alignment

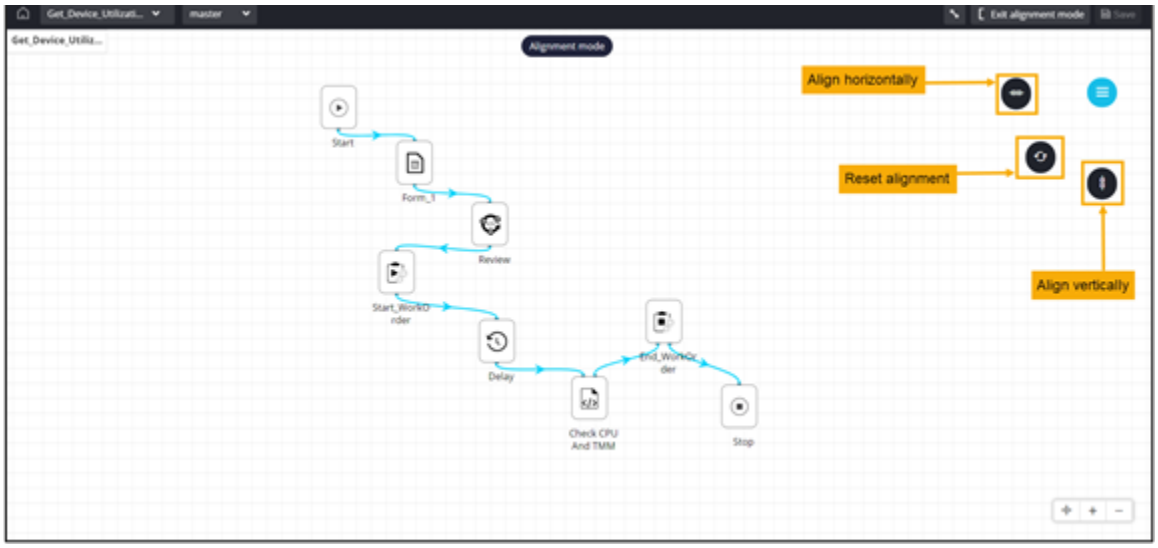
You can align a workflow either vertically or horizontally.

1. Go to the Workflow page.
2. Open a workflow.
3. To see alignment options, from the right upper corner of the screen, click .
4. From the options displayed, select **Auto align tasks**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



5. Click **Yes** in the **Confirmation** pop-up window.
6. Select the alignment for the workflow from the options available.

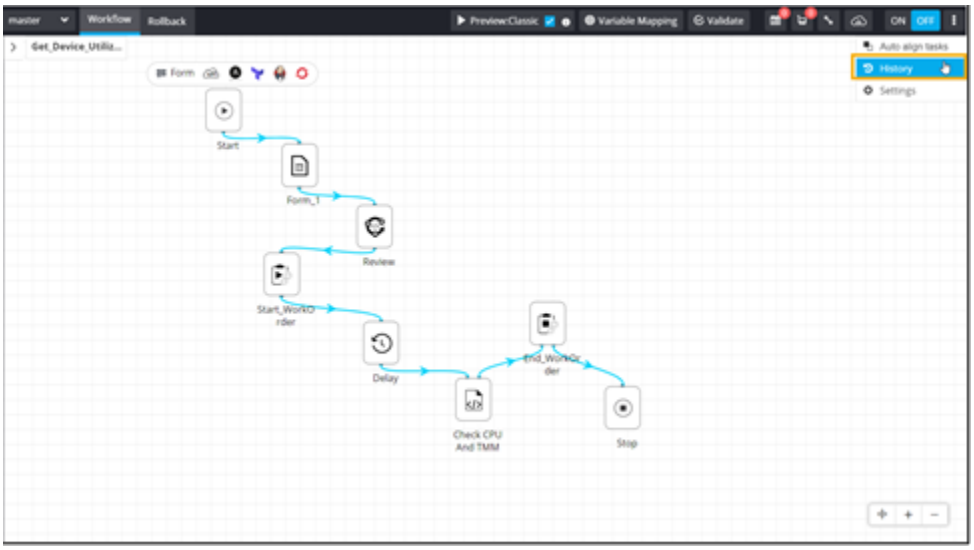


History

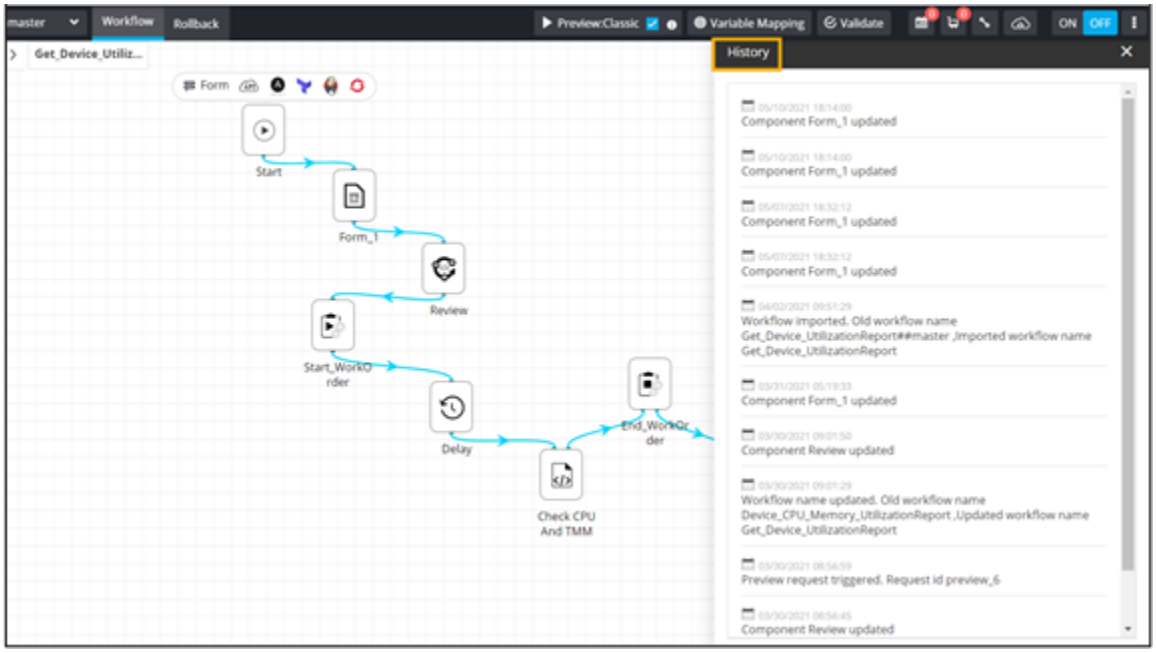
You can track the complete audit of changes made against a workflow.

1. Go to the Workflow page.
2. Open a workflow.
3. From the top right corner of the screen, click .
4. From the options displayed, select **History**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **History** window opens displaying the complete history of actions for this workflow.



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.

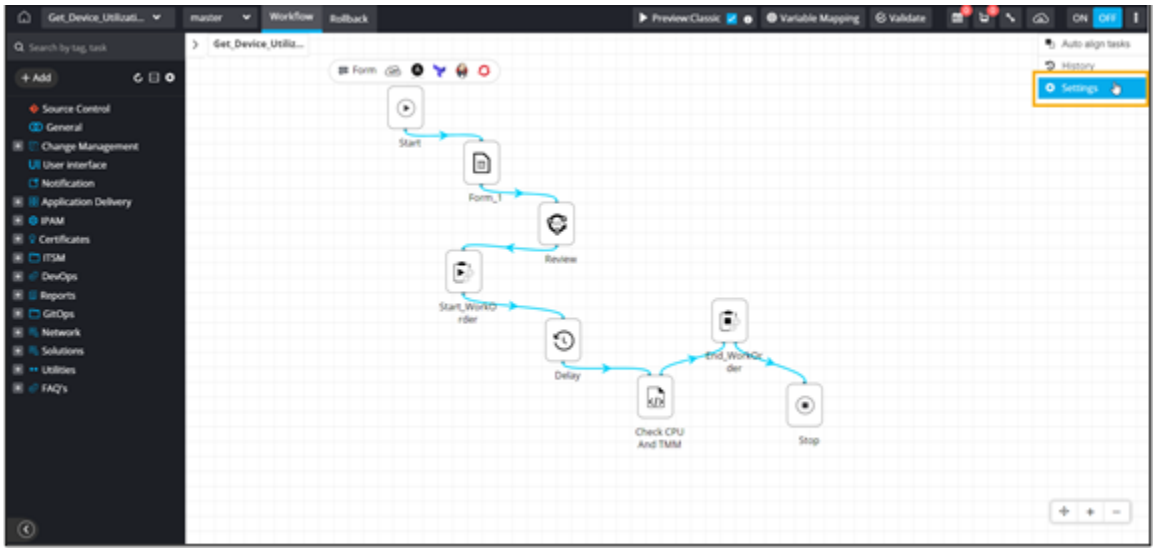
- Role based access control (RBAC) to assign workflows.
- Role based access control (RBAC) to assign workflow requests.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- 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:

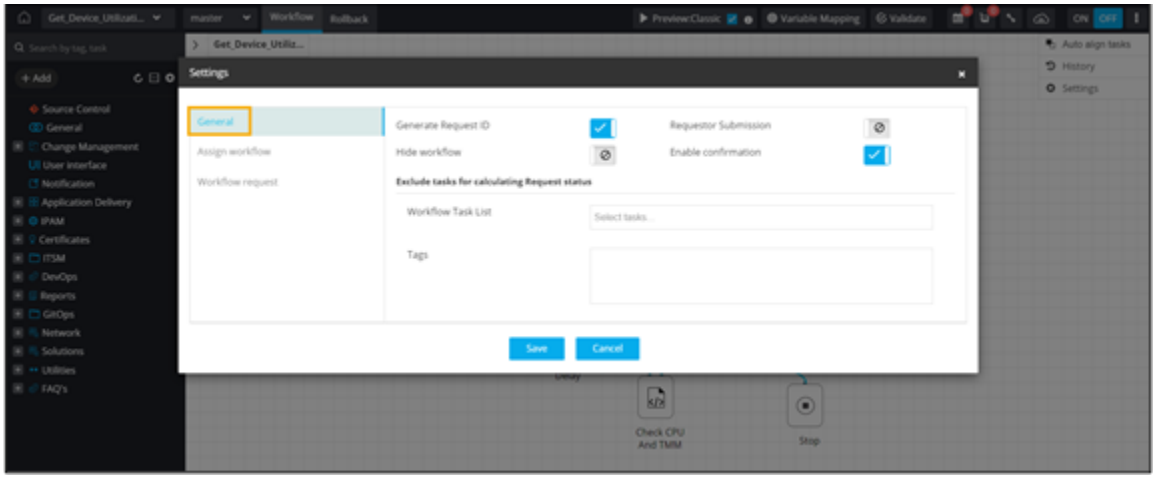
1. Go to the Workflow page.
2. Open a workflow.
3. From the top right corner of the screen, click .
4. From the options displayed, select **Settings**.



- General
- Assign Workflow
- Workflow Request

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

General



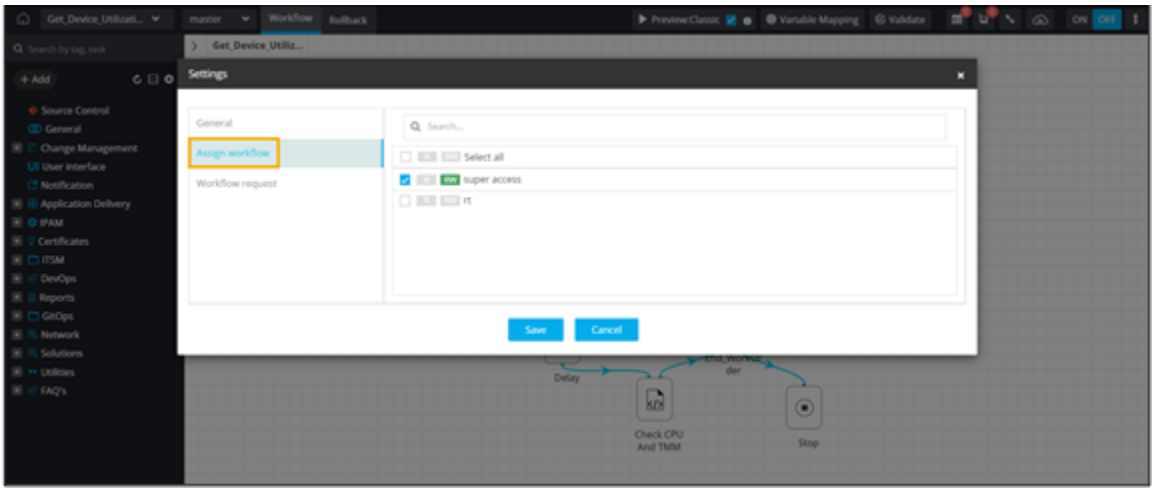
Under Workflow Settings, the following options are available under the General section:

- **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.

Assign Workflow

Under Workflow Settings, you can set up RBAC for workflows and assign workflows to specific users or roles.

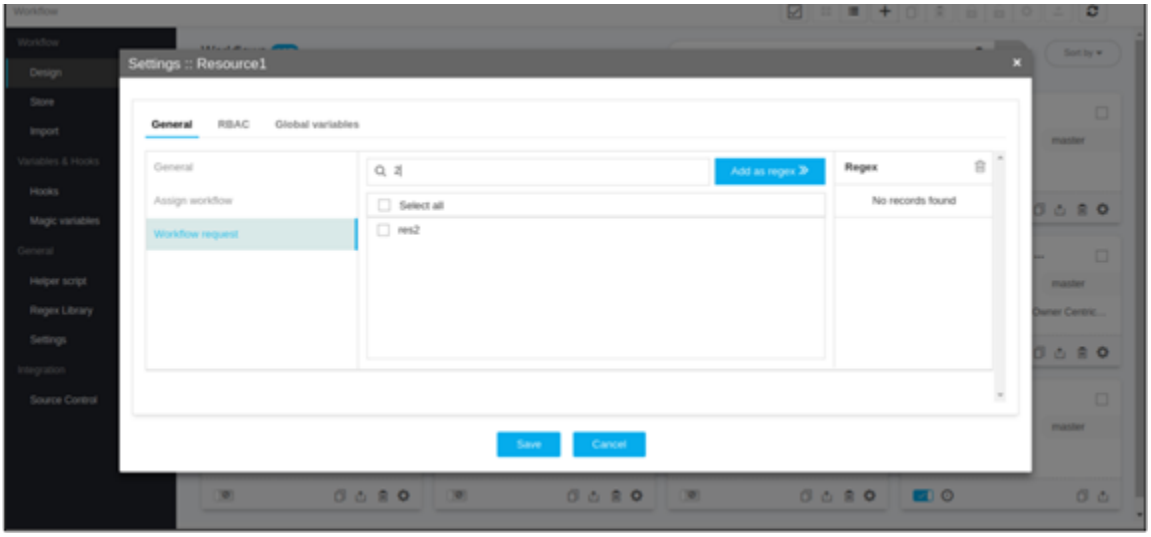
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Workflow Request

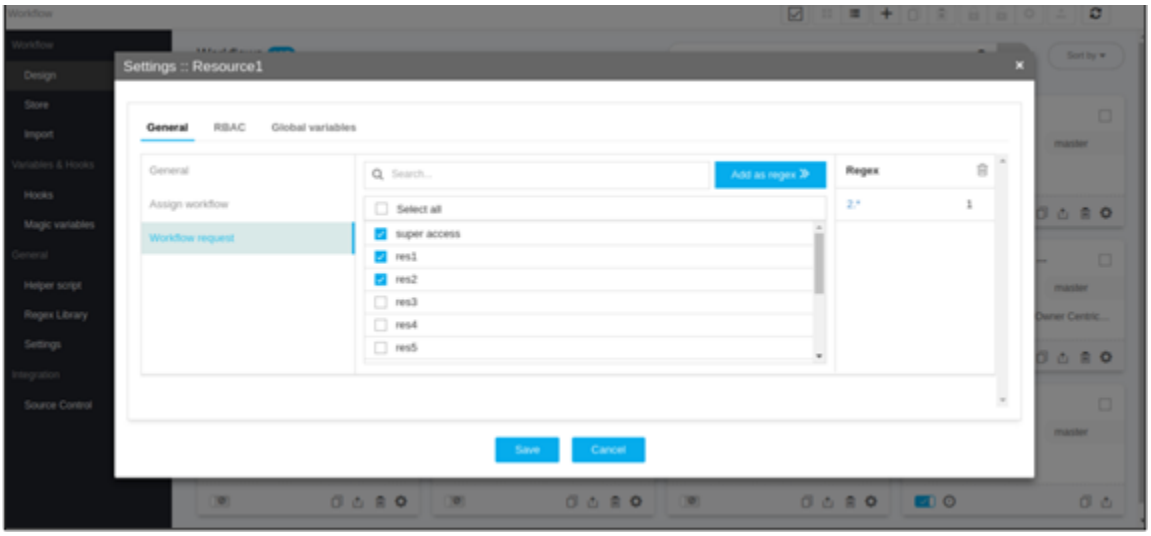
Under Workflow Settings, you can set up RBAC for workflow requests.

- 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 names can be used to select the list of all available resources.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- When trying to add the regex "*", a validation error will be triggered, resulting in the disabling of the "Add as regex" option.
- Provision to delete the regex.

Validating a Workflow

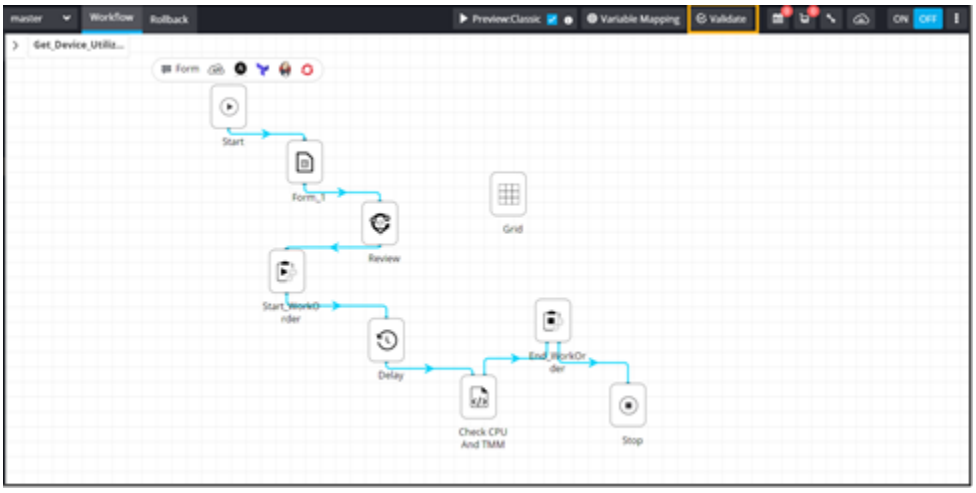
Once a workflow is designed, users can validate the workflow and its elements. Basic workflow validation includes:

- Tasks with no relevant connectors associated.
- Missing tasks which are dependent on other workflow tasks.
- Invalid links or connections.

To validate a workflow:

1. Design a workflow.
2. In the design space, from the command bar on the top, click **Validate**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



If the workflow is valid, a pop-up stating

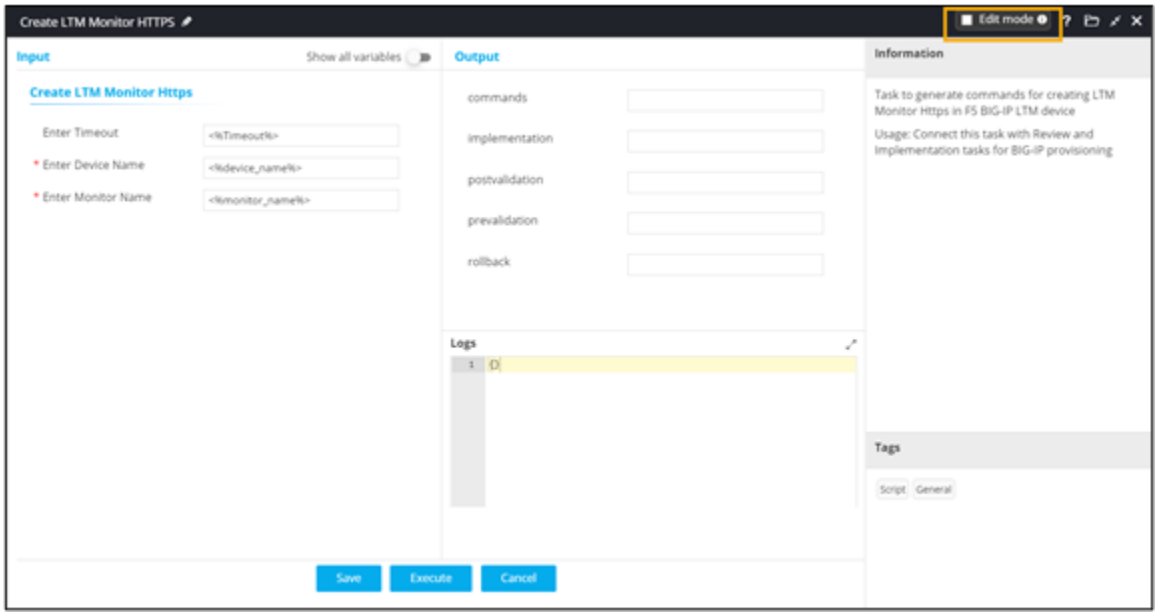
(Valid workflow) is displayed.

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, deselect the **Edit mode** checkbox.
   In the Citizen mode of the task window, variables referred in the script are listed under **Input**, while the values declared as Global Variables are listed under **Output**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

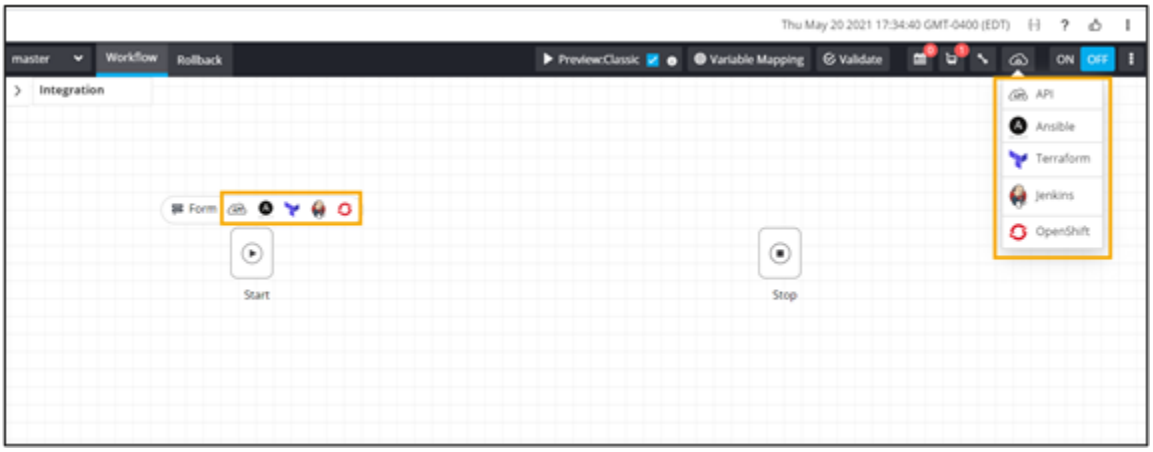


3. To switch back to Edit mode, select the **Edit mode** checkbox.

Northbound 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, and Postman.



Copyright © 2025 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

Postman - Visual Workflow Northbound Integration

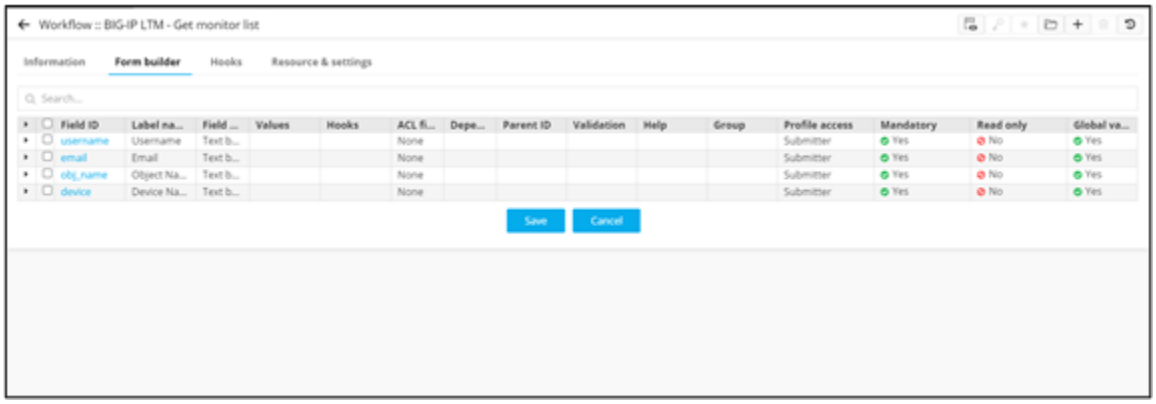
This integration enables dynamic API generation from a third-party payload generator such as Postman, which can be used to design custom automation workflows.

- 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 required form fields.



4. From the **General** section, drag and drop a **Script** task.
5. From the **User Interface** section, drag and drop an **Email** task.
6. To generate an API for this workflow, click API icon above the **Start** task.
   The URL, HTTP method, Query params, Payload, and the Header for the API are displayed.

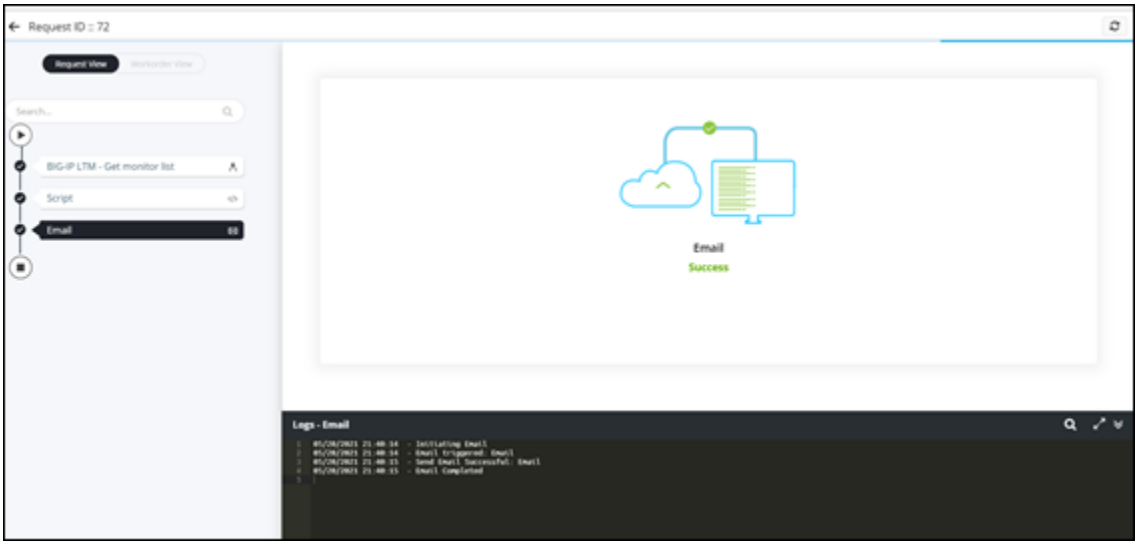
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: The API can also be generated as cURL or Python.

7. To download this API, form the top right corner of the API window, click **(Download)** icon.
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. Navigate to the **Request :: All** page and click on the **Request ID** generated in Postman.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Workflow is executed successfully.



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,

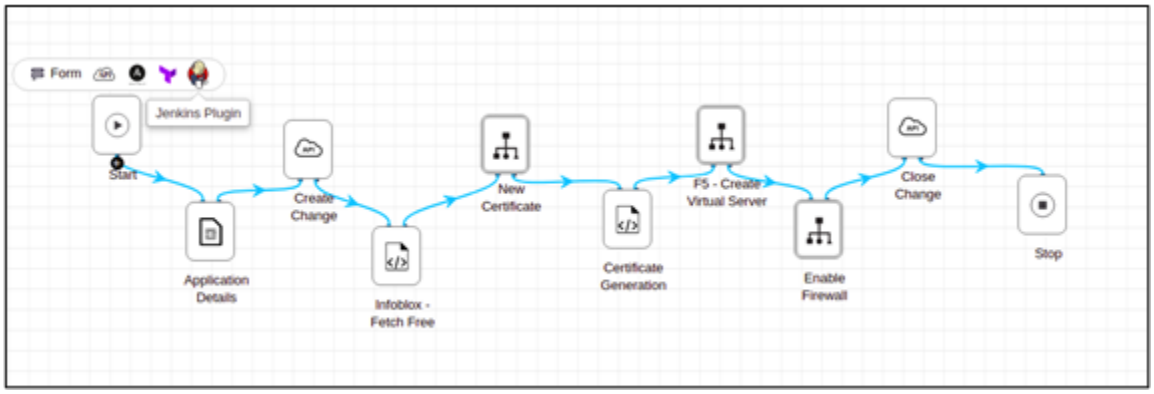
- Download a Jenkins plugin from Visual workflow
- Install the Jenkins plugin and set it up
- Execute Jenkins job to trigger workflows
- Create Pipelines
- Build the Application Service Framework via Jenkins.

- Prerequisites
- Global Configuration
- Creating a Freestyle Job
- Creating a Pipeline Job
- Jenkins pipeline for Application provisioning

Copyright © 2025 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:

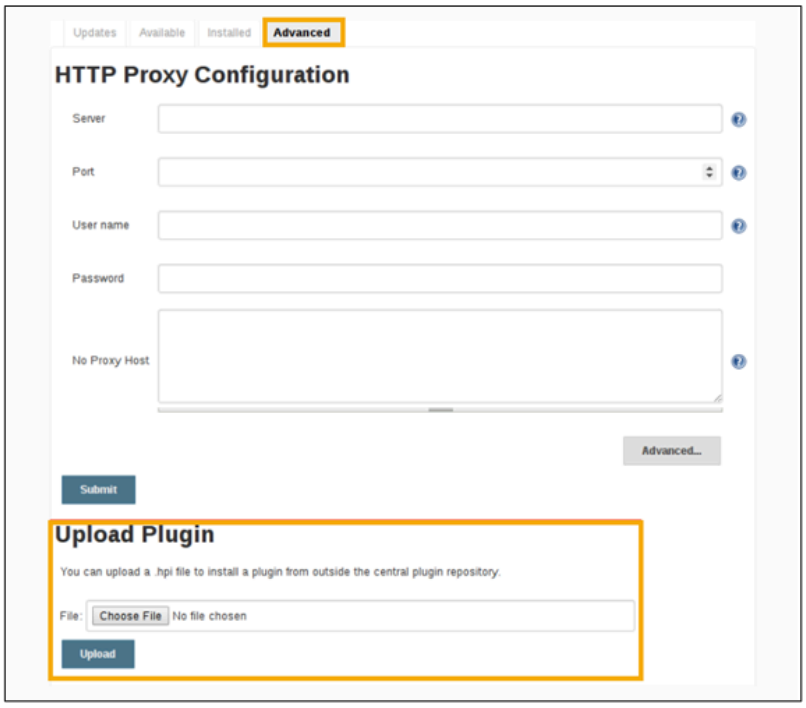
- AppViewX.hpi (the plugin to be installed in the Jenkins)
- Jenkins and AppViewX Integration Guide <.html>
- Jenkins and AppViewX Integration Guide <.pdf>.

To install the AppViewX plugin in Jenkins:

1. Go 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. **Upload** the plugin file.



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 of hostname, port, username, and password.

Creating a Freestyle Job

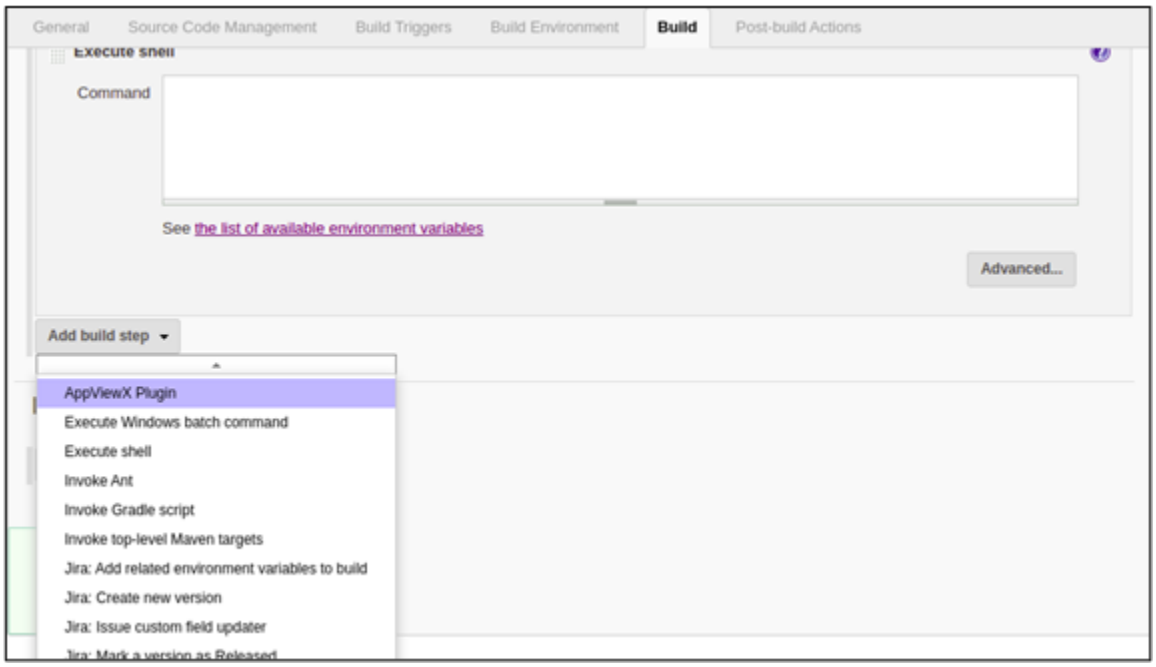
The freestyle build job is a highly flexible and easy-to-use option. It is a repeatable build job which contains steps and post-build actions.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

To create a freestyle job in Jenkins:

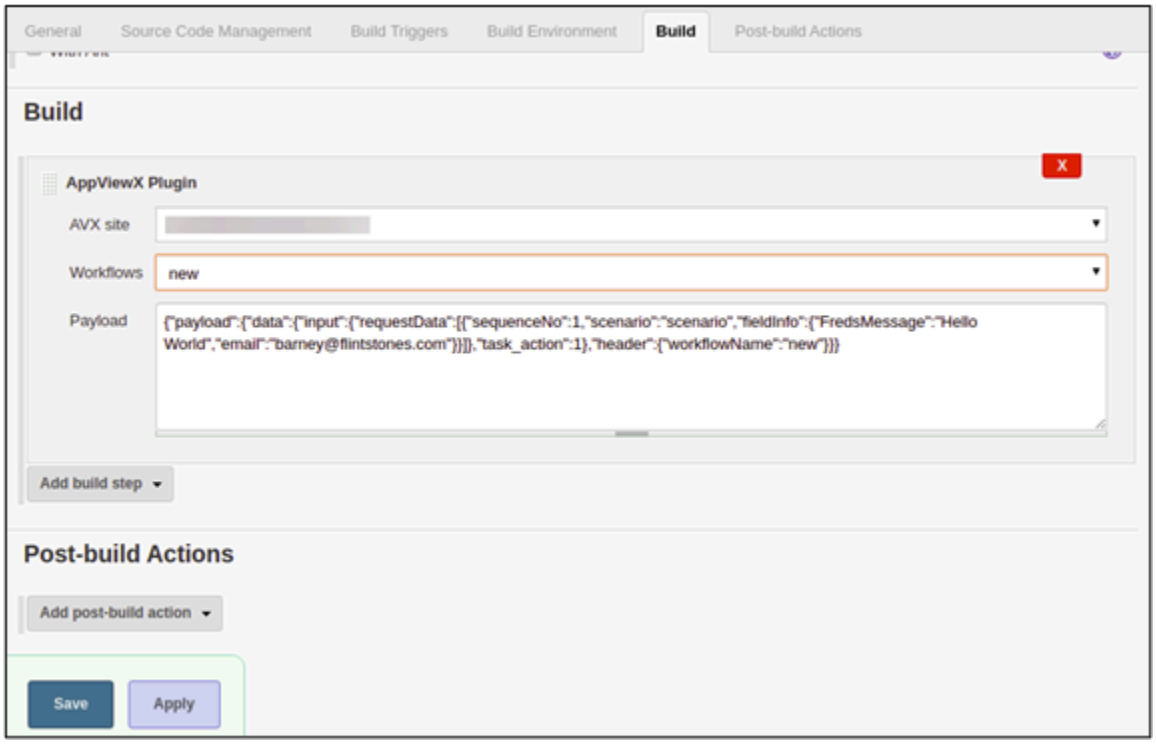
1. Click **New Item** in the upper left corner of the admin console.
2. Enter the name of the project.
3. Select **Freestyle Project**.

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 © 2025 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.



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):

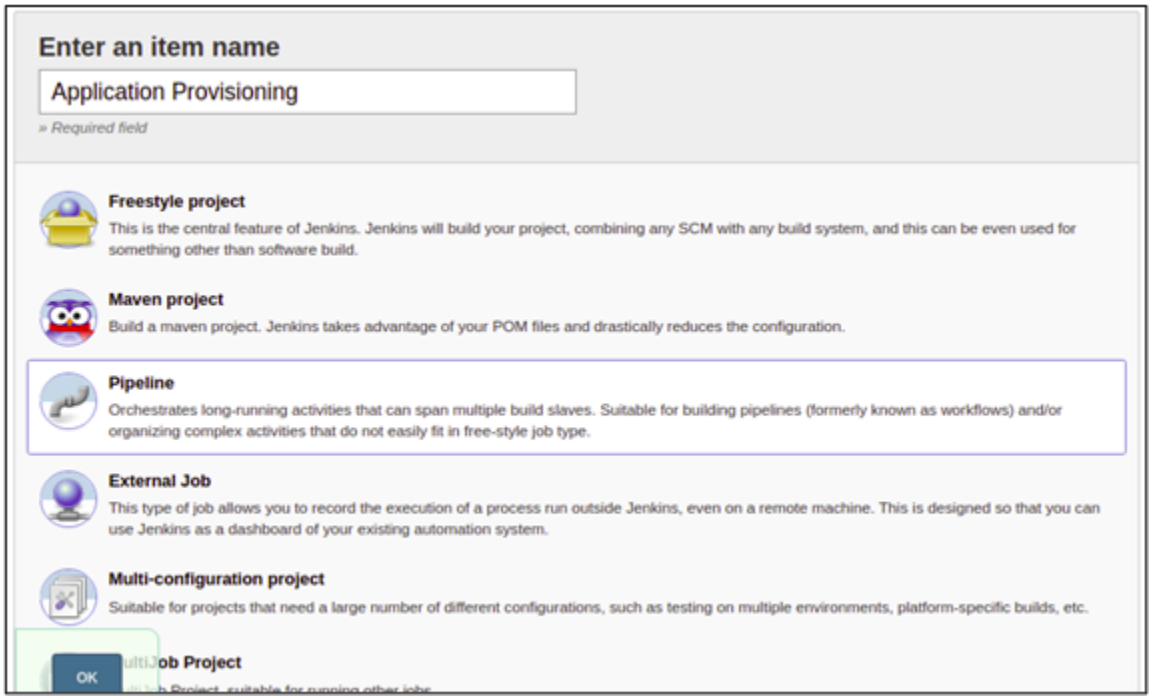
```
appViewXCheckStageStatus(String siteName, String requestId, String startTask, String endTask)
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Jenkins pipeline for Application provisioning

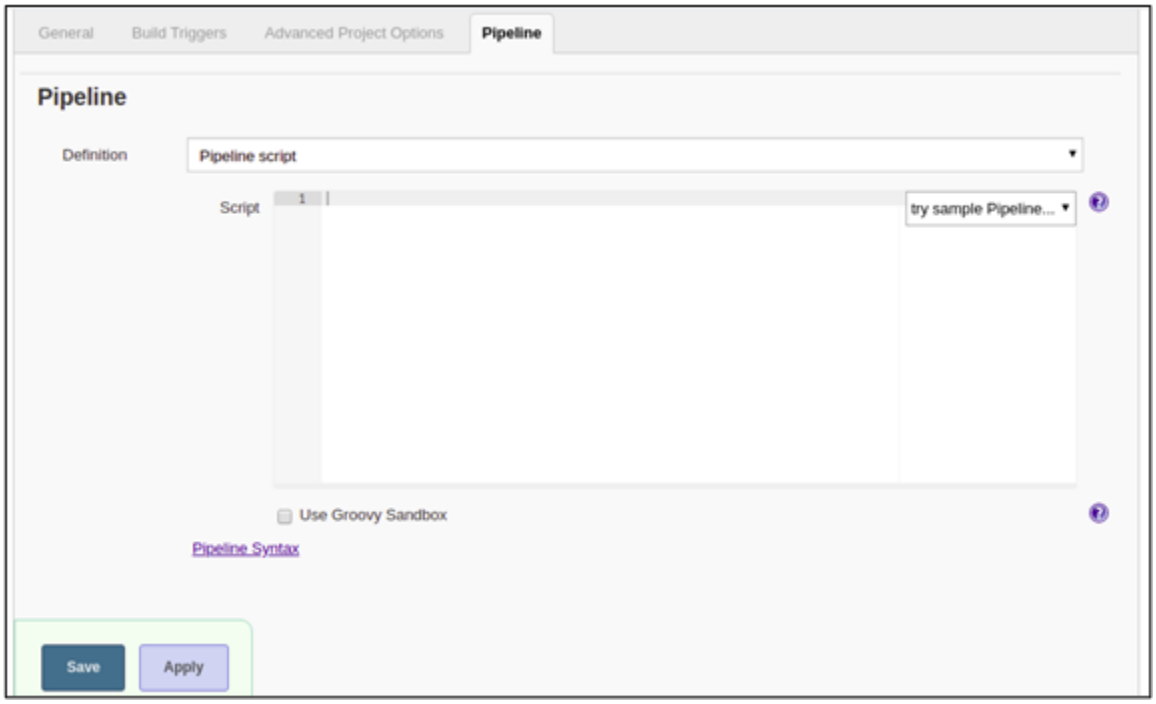
To create a pipeline job in Jenkins:

1. Click **New Item** in the upper left corner of the admin console.
2. Enter the name of the project.
3. Select **Pipeline**.



4. Under **Pipeline**, enter the pipeline script, refer to the example pipeline script given here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



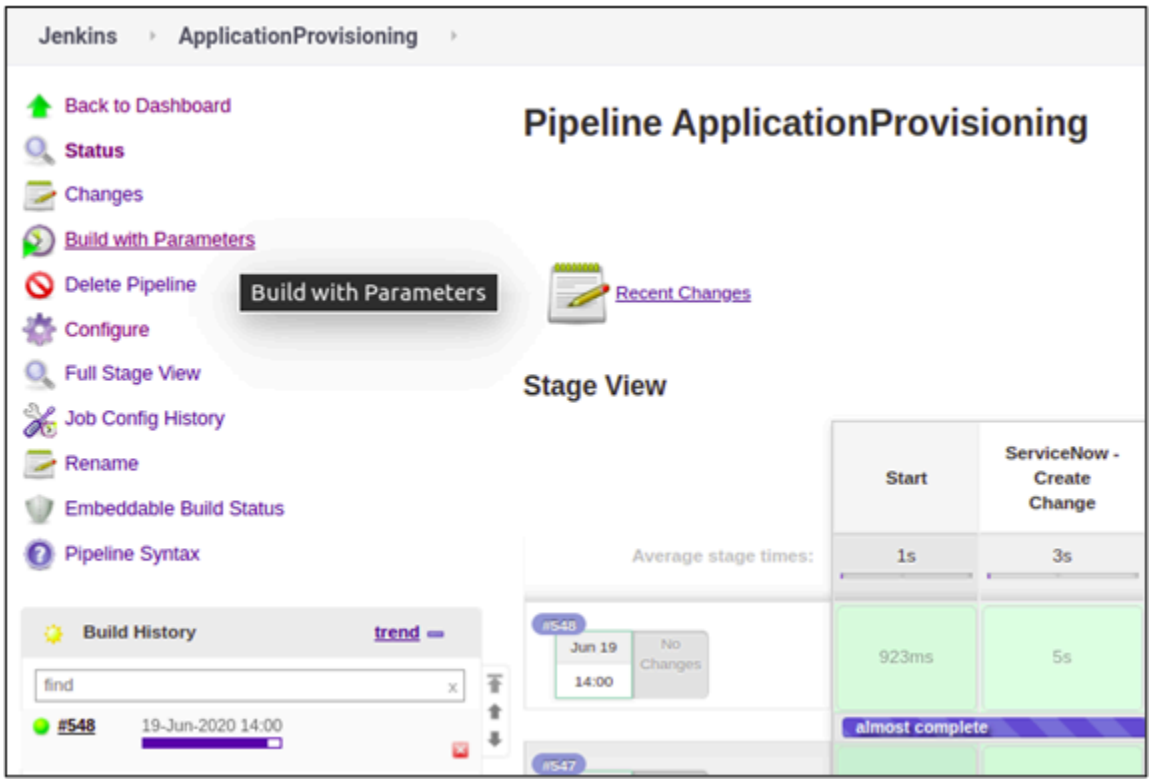
Example Pipeline Script:

```
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:
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

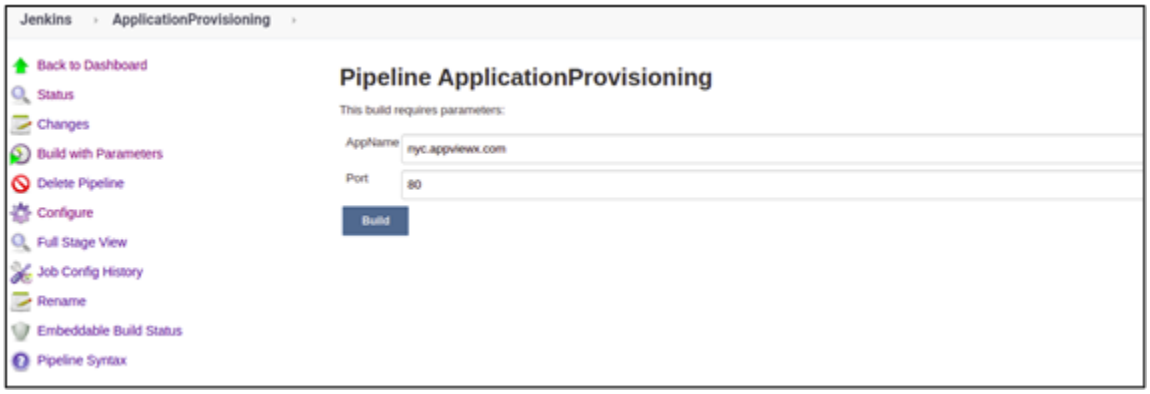
```
"script_1:Infoblox - Fetch Free IP's") }
stage('Certificate Generation') { appViewXCheckStageStatus
(siteName:"admin@https://192.168.142.172:5300", requestId: "${RequestId}",
startTask: "", endTask: "script_2:Certificate Generation") }
stage('F5 BIG IP Provisioning')
{ appViewXCheckStageStatus(siteName:"admin@https://192.168.142.172:5300",
requestId: "${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**.
6. On the stage view page, where the Pipeline build will be executed, choose **Build with Parameters** option.



7. Enter the required inputs to run the workflow and click **Build**.

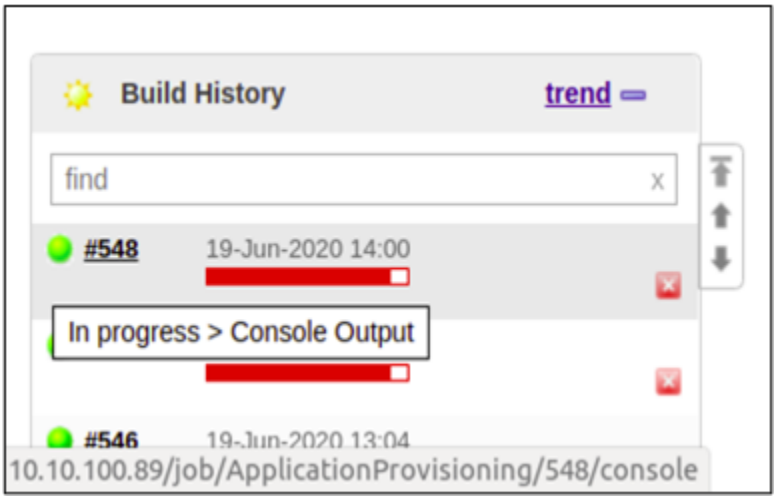
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



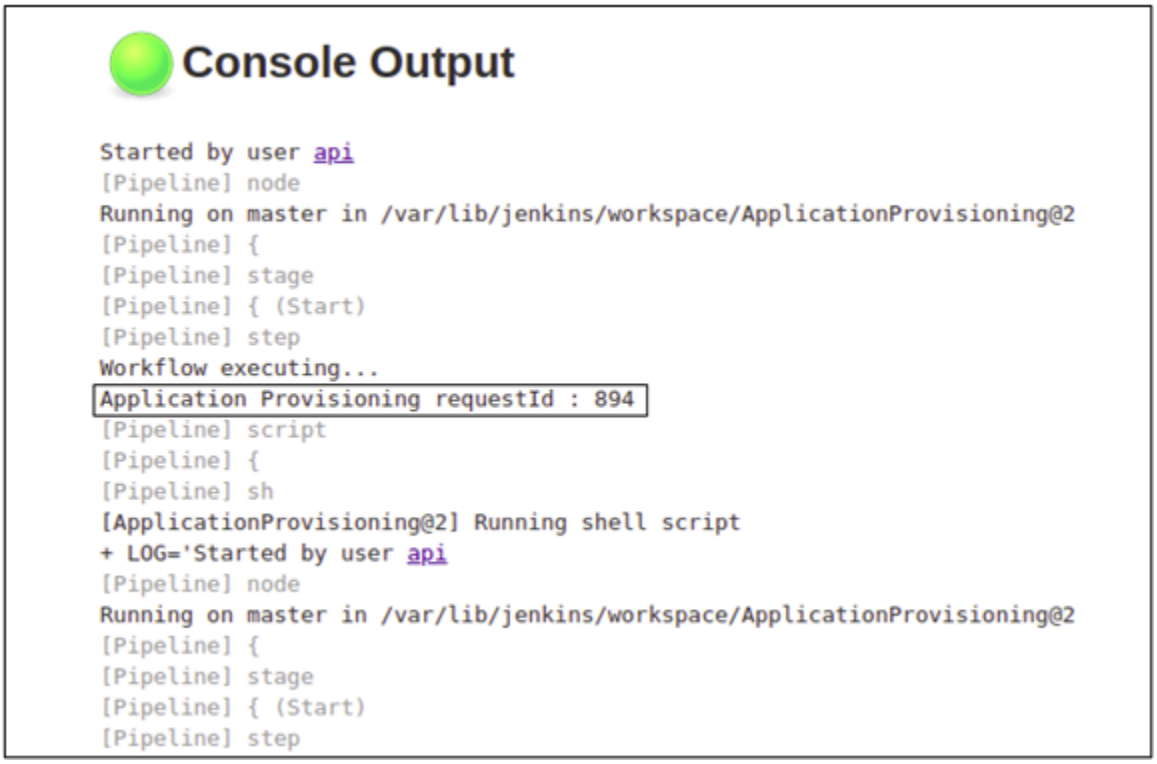
8. The stage view and console output can be seen when the pipeline is executed.
   - Pipeline stages

   - Console Output: Select the globe near the build number under Build History to see the outputs.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- The request Id of the workflow is shown in the console logs.

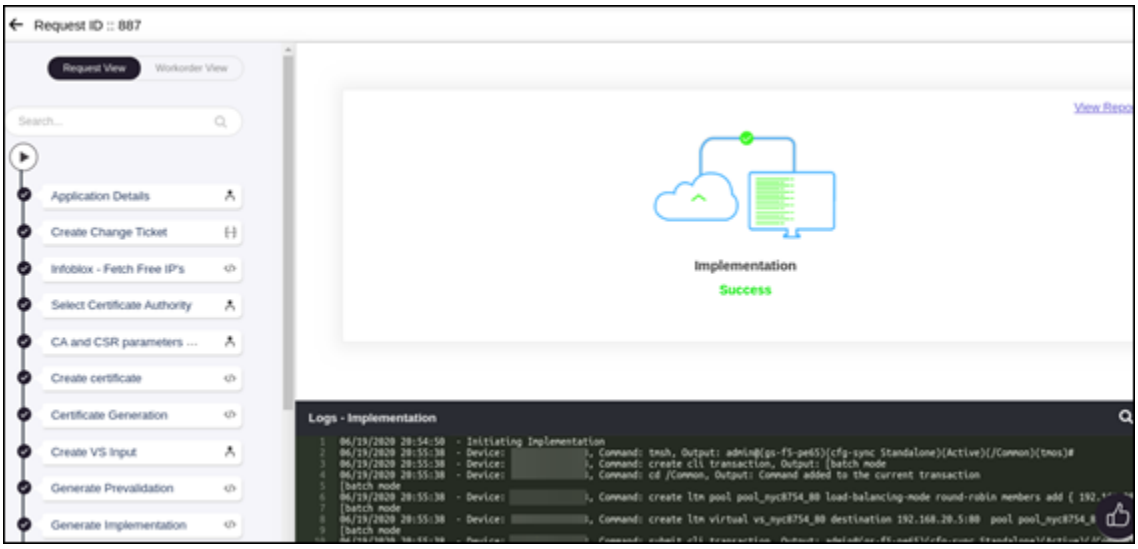


9. The task name, task logs and task status for each task in the workflow also shown in the console logs.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

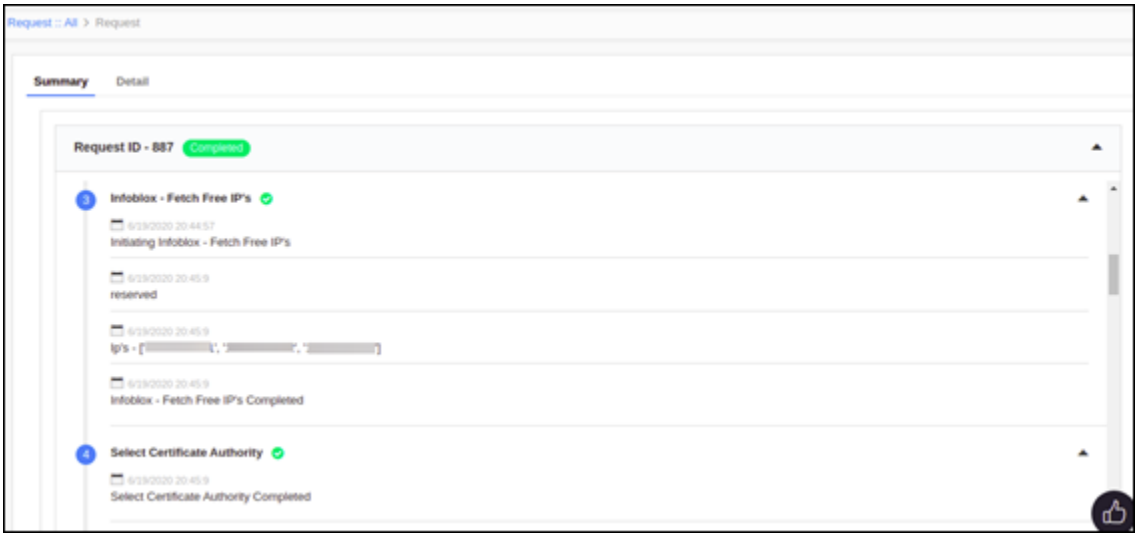
```
task name : Infoblox - Fetch Free IP's
Initiating Infoblox - Fetch Free IP's
reserved
Ip's - [                l', '                ', '                ']
Infoblox - Fetch Free IP's Completed
task status : Success


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. On the Request :: All page, click on the **Request ID** of any workflow to see the execution of each task in the workflow.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

11. 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.

1. Design a new workflow.
2. In the design space, to download the .zip file, click above the **Start** task.

> **Tip:** You can also click on the **Service Orchestration** icon in the command bar on the top, to download the <.zip> file.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Extract the <.zip> file.
4. From the extracted <.zip> file, copy the <workflowname>.tf file to the installation path of AppViewX Terraform provider.
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"
```

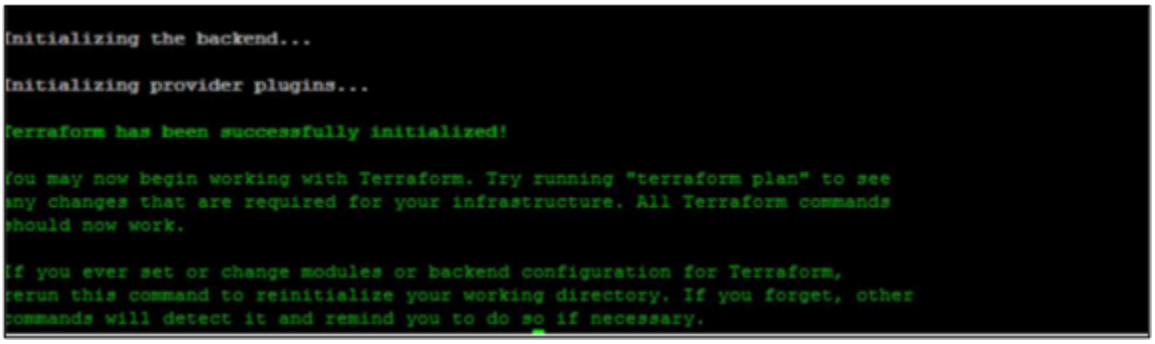
7. Enter workflow request details under the payload section.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

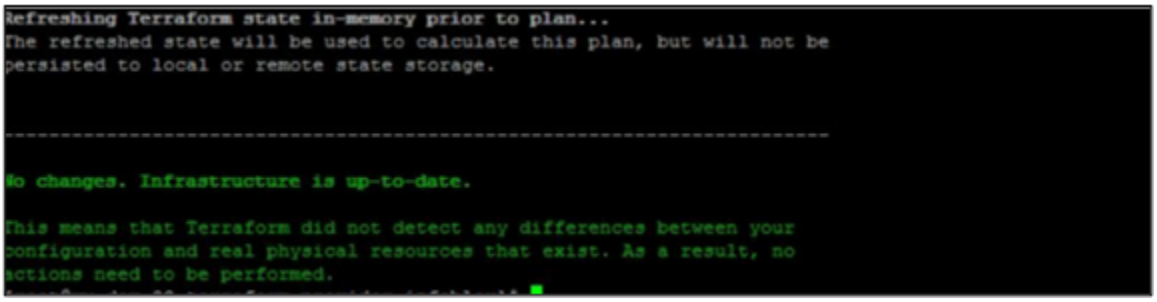
Sample:

```
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



   - terraform apply.

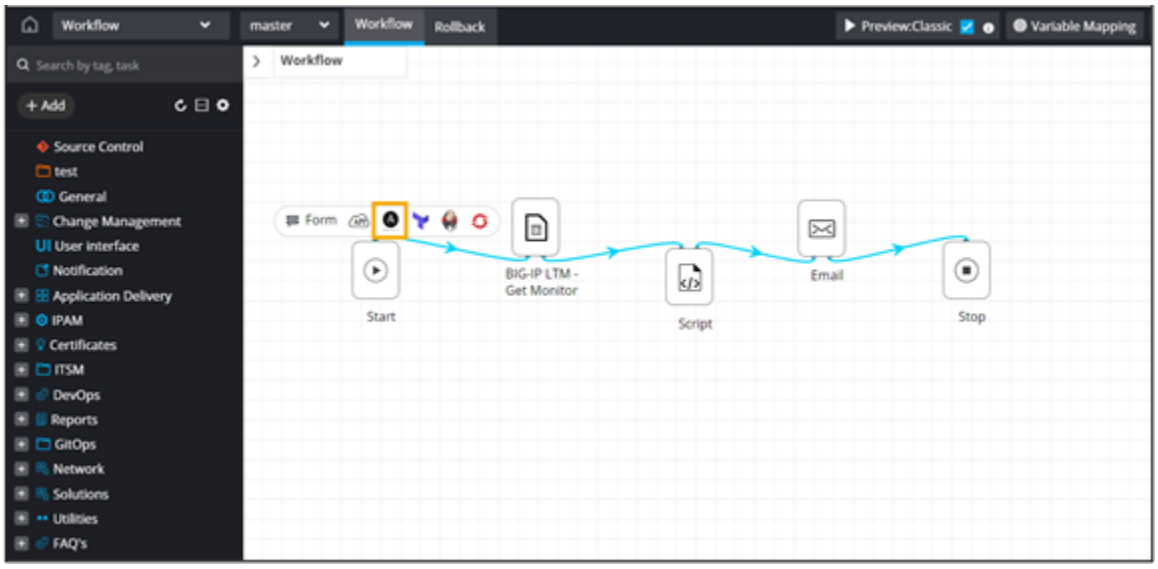
Ansible - Visual Workflow Northbound Integration

You can automate network configurations through Ansible as the Northbound.

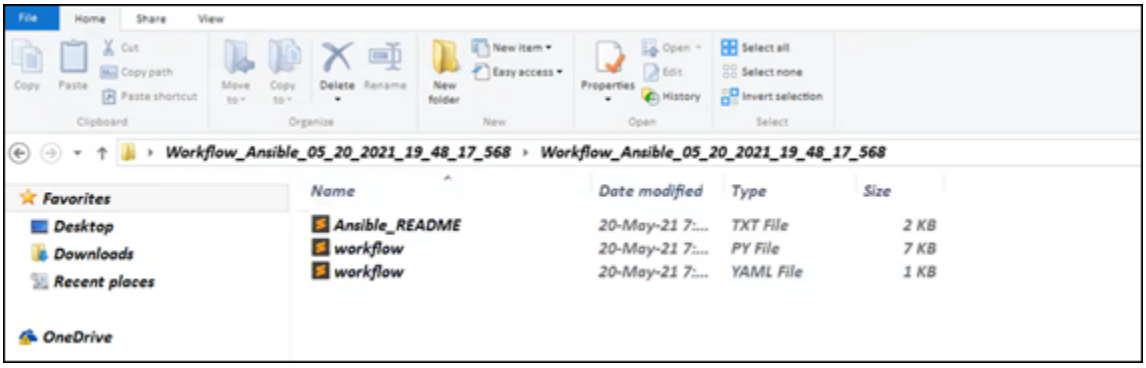
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

To trigger a workflow by creating an Ansible playbook:

1. Design a workflow.
2. Connect all workflow tasks and enable the workflow.
3. To download the Ansible <.zip> file, click icon above the **Start** task.



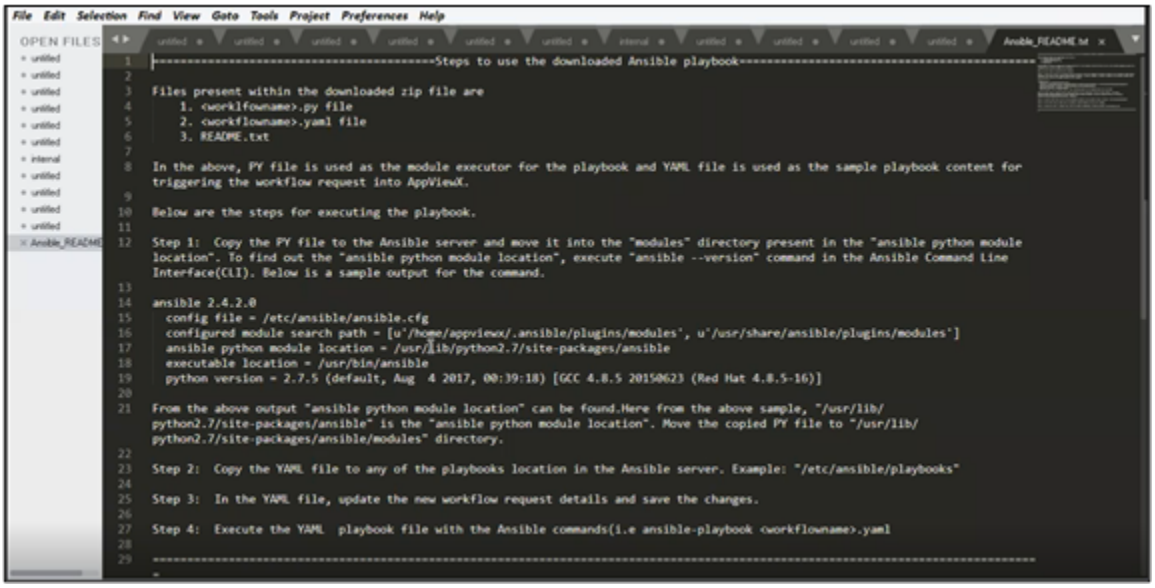
4. Extract the files.



The downloaded <.zip> folder contains the following files:

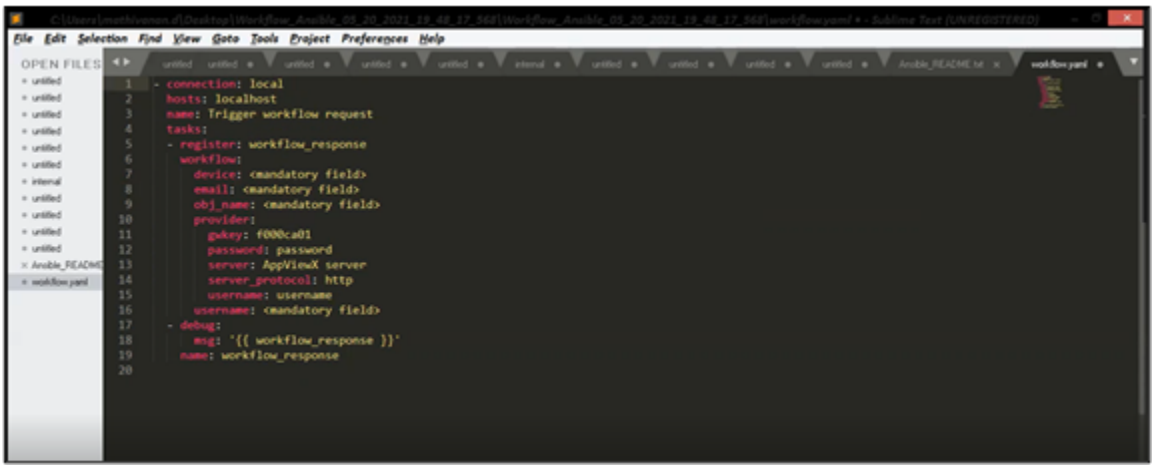
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Ansible_README txt file - This contains the steps to use the downloaded Ansible playbook.



```
==========Steps to use the downloaded Ansible playbook==========

Files present within the downloaded zip file are
    1. <workflowname>.py file
    2. <workflowname>.yaml file
    3. README.txt

In the above, PY file is used as the module executor for the playbook and YAML file is used as the sample playbook content for
triggering the workflow request into AppViewX.

Below are the steps for executing the playbook.

Step 1:  Copy the PY file to the Ansible server and move it into the "modules" directory present in the "ansible python module
location". To find out the "ansible python module location", execute "ansible --version" command in the Ansible Command Line
Interface(CLI). Below is a sample output for the command.

ansible 2.4.2.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/appviewx/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Aug  4 2017, 00:39:18) [GCC 4.8.5 20150623 (Red Hat 4.8.5-16)]

From the above output "ansible python module location" can be found.Here from the above sample, "/usr/lib/
python2.7/site-packages/ansible" is the "ansible python module location". Move the copied PY file to "/usr/lib/
python2.7/site-packages/ansible/modules" directory.

Step 2:  Copy the YAML file to any of the playbooks location in the Ansible server. Example: "/etc/ansible/playbooks"

Step 3:  In the YAML file, update the new workflow request details and save the changes.

Step 4:  Execute the YAML  playbook file with the Ansible commands(i.e ansible-playbook <workflowname>.yaml

==========
```

- <workflowname>.py - Workflow Python file (Ansible executor file)
- <workflowname>.yaml - Workflow YAML file



```
- 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
```

In the above, the Python (.py) file is used as the module executor for the playbook and the YAML file (.yaml) is used as the sample playbook content for triggering the workflow request into AppViewX.

5. To execute the playbook, copy the Python <.py> file to the Ansible server and move it into the modules directory present in the ansible python module location.
6. To find out the ansible python module location, execute the ansible --version command in the Ansible Command Line Interface (CLI).

   Here is a sample output for the command:

```
ansible 2.4.2.0

 config file = /etc/ansible/ansible.cfg
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
configured module search path = u'/home/appviewx/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /usr/bin/ansible

python version = 2.7.5 (default, Aug  4 2017, 00:39:18) [GCC 4.8.5 20150623 (Red Hat 4.8.5-16)]
```

In this sample command, the ansible python module location is `/usr/lib/python2.7/site-packages/ansible`

7. Copy the YAML file to any of the playbooks location in the Ansible server. For example: `/etc/ansible/playbooks`
8. In the YAML file, update the new workflow request details and save the changes.
9. Execute the YAML playbook file with the Ansible commands - `ansible-playbook <workflowname>.yaml`

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.

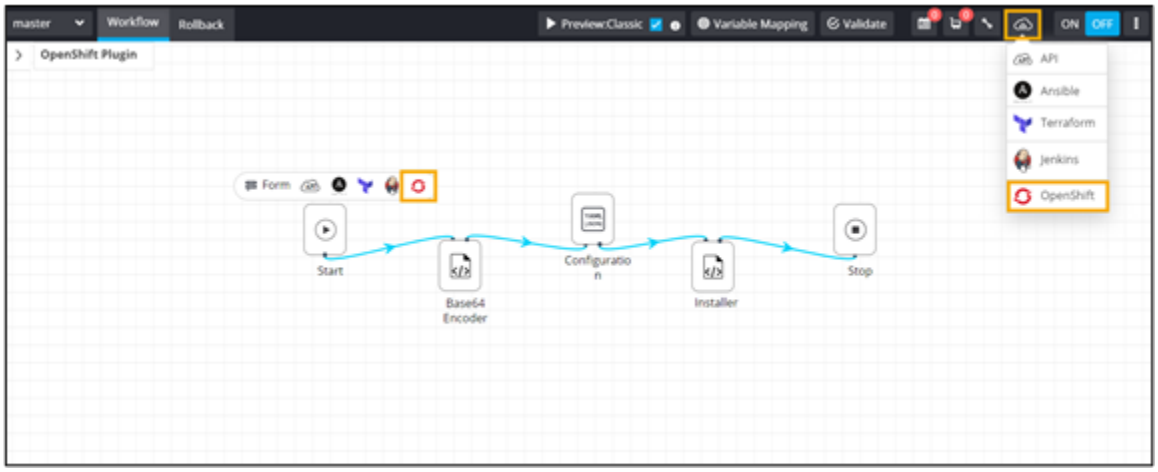
- Offers a Kubernetes container platform to manage hybrid cloud deployments.
- Allows you to connect application deployed in OSE Container Platform to a variety of service brokers.
- OSB API manages applications defined by AppView catalogs.
- Enables PaaS Orchestration of application and PKI services.
- Accelerated application deployment.

To facilitate Openshift PaaS Orchestration:

1. Design a new workflow.
2. To download the OSB Plugin manually, click icon above the **Start** task.

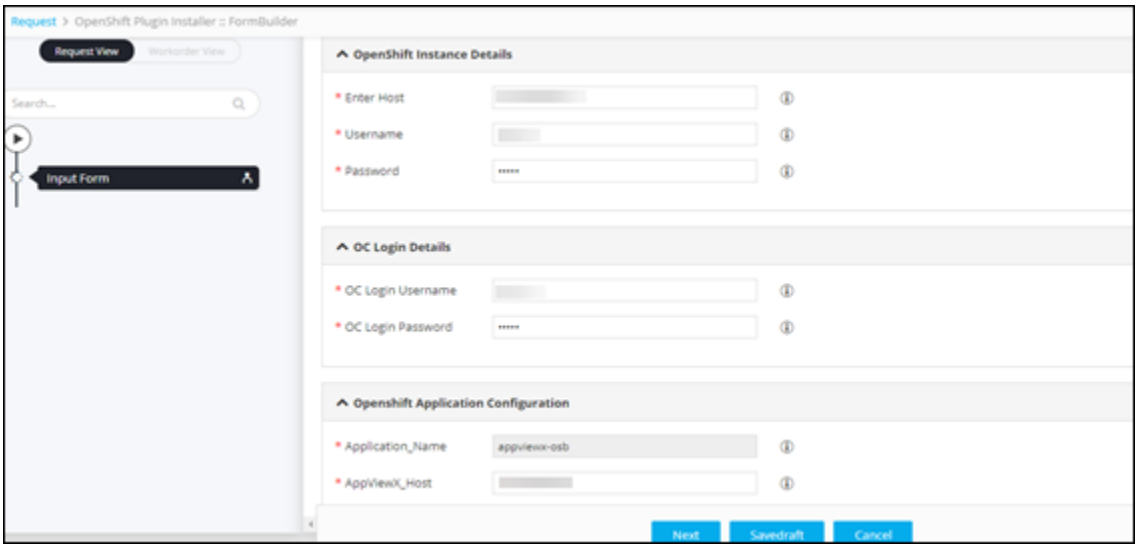
> **Tip:** You can also click on the **Service Orchestration** icon to install the plugin.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



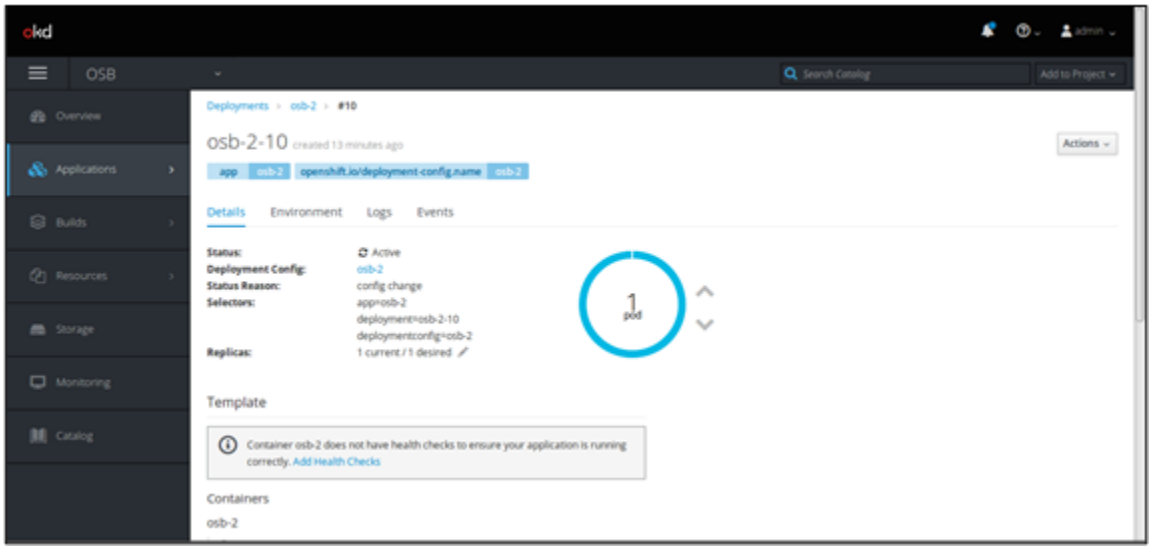
3. To install the AppViewX OSB plugin automatically, trigger the OpenShift Plugin Installer workflow from the **Request :: View/Run** page.

   The workflow execution page is displayed with the workflow inputs requested at the first stage.
4. Enter the OSB details in the fields.



5. To proceed with plugin installation, click **Next**.
6. Open the AppViewX Open Service Broker.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



7. View all automation catalogs in OpenShift once the AppViewX broker is installed.
8. To provision application and security services through AppViewX, click on any Automation catalog.

> **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.

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	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>`

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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)

  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},"{"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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

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}}

Service Requests

The Service Requests console allows you to execute workflow(s). Workflows can be executed manually or can be scheduled. The **Request :: Overview** page 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.

- Configuring RBAC for Workflow Request Stage View
- Accessing Service Requests
- Request :: Overview
- Custom Reports
- Request :: View/Run
- Scheduled Jobs
- Request :: All

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Request :: Open
- Request :: Closed
- Request :: Failed
- Request :: Assigned
- Request :: Audit Logs
- Configuring RBAC for Archive and Restore Requests
- Archive Requests
- Restore Archives
- Preferences
- Service Request Actions

Configuring RBAC for Workflow Request Stage View

You can configure permissions to allow or restrict users to access the stage view when the workflow request is executed.

To hide the stage view of workflow stages in workflow execution:

1. Navigate to the **Role** page in the **Platform** module.
2. Select the role for which you want to enable access. For example, **admin**.
3. Under **Authorized functions**, expand **Automation** > **Service Requests** > **User Preference**.

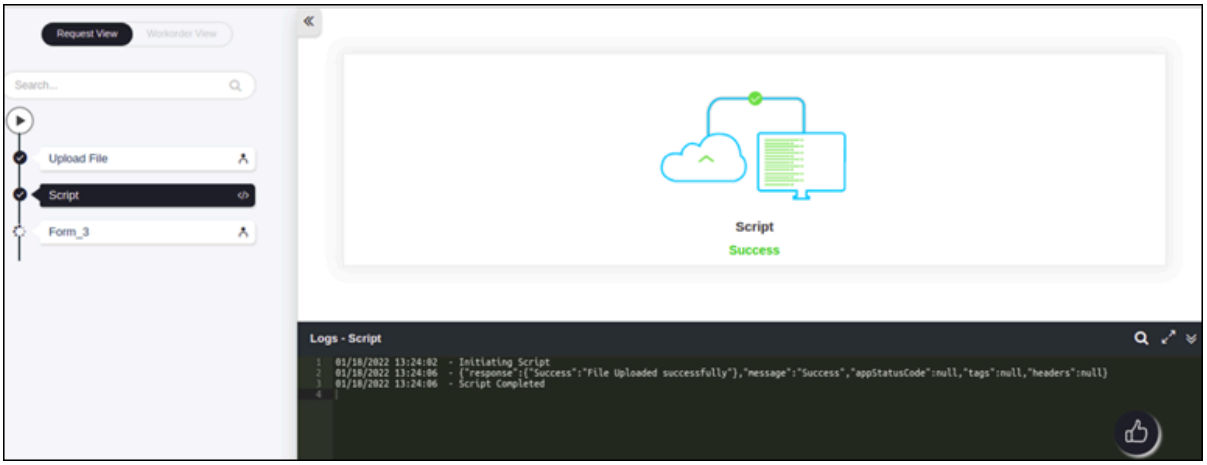
> **Note:** If you are using the old menu, under the **Authorized Functions** tab, expand **Request** > **Workflow**.

4. Under **User Preference**, ensure that the **Show request stage** checkbox is **not** selected.
5. Click **Save**.
   The stage view is not displayed for the user and only the details of the current task are displayed.

6. To display the stage-wise view in workflow execution, under **User Preference**, select the **Show request stage** checkbox.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

List of tasks is displayed on the left side of the screen.



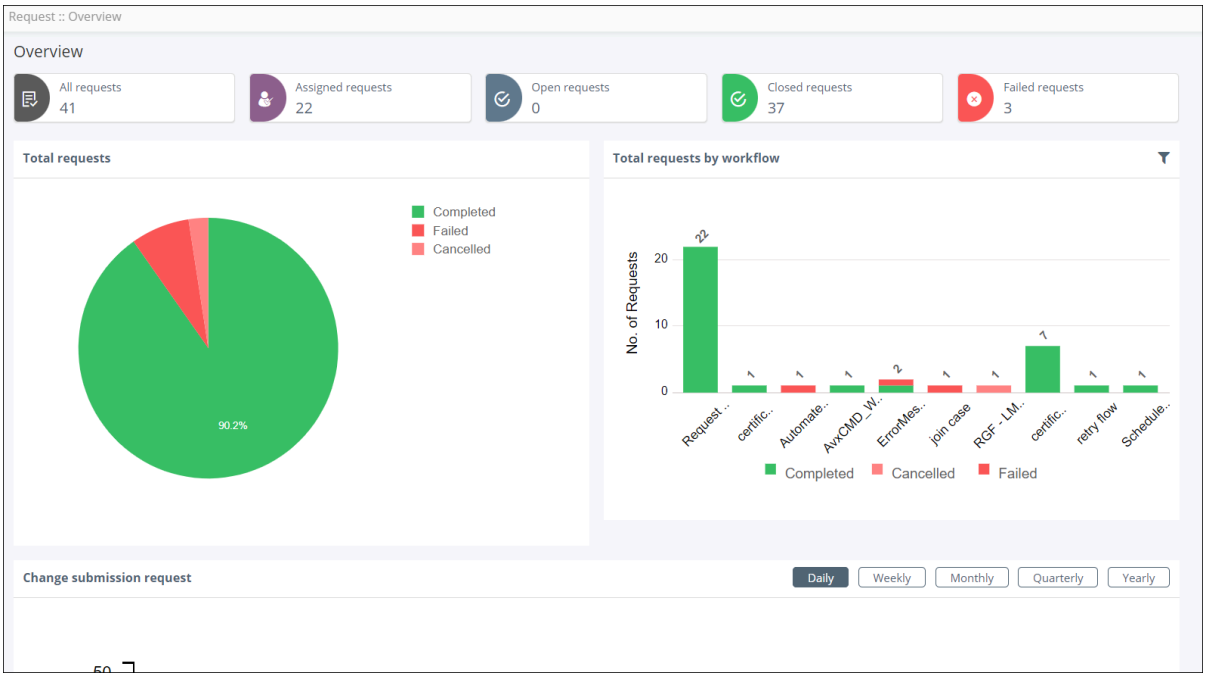
Accessing Service Requests

New Menu	Old Menu
From the main navigation menu, select **Automation**. The Request :: Overview page is displayed.	From the main navigation menu, select **Request**. The Request :: Overview page is displayed.

Request :: Overview

The **Request :: Overview** page provides a one-stop view of all workflow requests by status - opened, closed, failed, assigned, and total workflow requests. It allows you to generate dynamic reports on the status of the workflow requests.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Click on any section of the chart or metric and navigate directly to the inventory. Following are the possible statuses for a workflow request:

Status Descriptions

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.

Total requests

The **Total Requests** pie chart displays all the service requests as per their status.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Total requests by workflow

The **Total Requests by workflow** bar chart displays the number of requests triggered for each workflow along with their statuses

Change Submission Request

The **Change submission requests** line chart helps you 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.

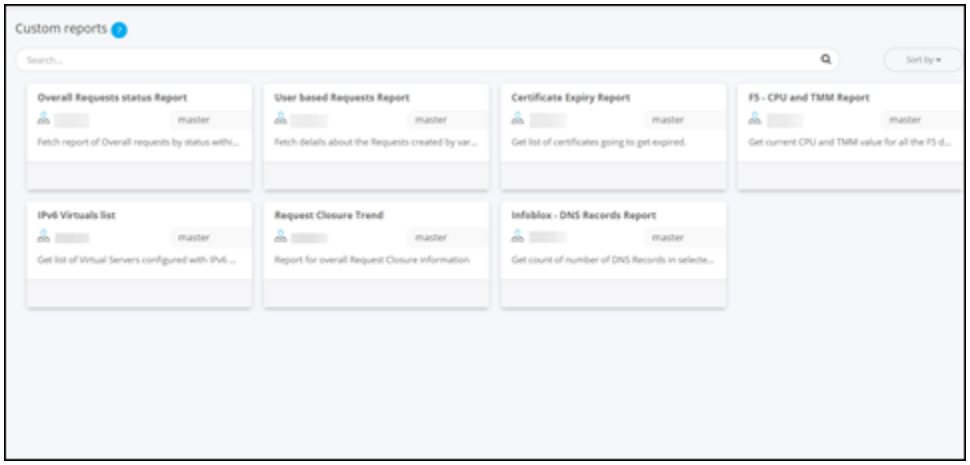
Custom Reports

This section allows you 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• Provision to execute/run a custom report workflow.



• Triggering Custom Reports

Triggering Custom Reports

To execute a custom report workflow:

1. On the Request :: Overview page, from the menu on the left, click **Custom Reports**.
2. To execute any report workflow on the **Custom Reports** page, hover your mouse over the workflow and click **(Trigger)** icon.
   The workflow execution page is displayed with the workflow inputs requested at the first stage.

> **Note:** The inputs requested and workflow execution steps will vary as per the workflow that is triggered.

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.

• Triggering a Workflow
• Scheduling a Workflow

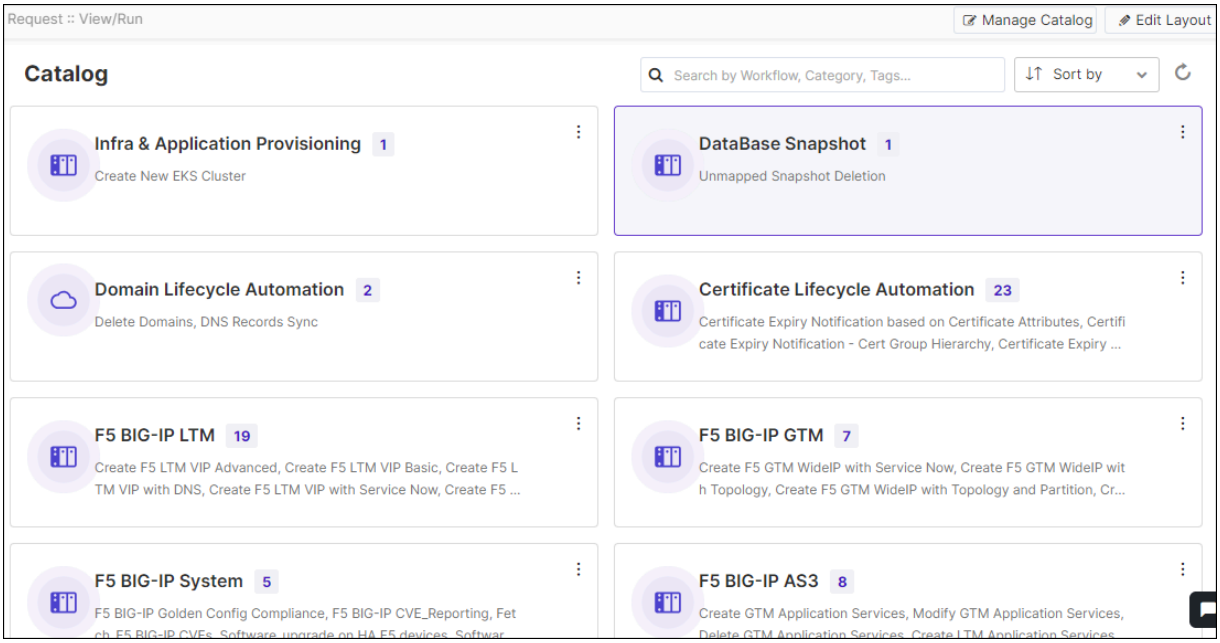
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Editing the Workflow Catalog Layout
- Editing Subcategories and Workflows within a Catalog

Triggering a Workflow

To trigger a workflow from the Workflow **Request :: View/Run** page:

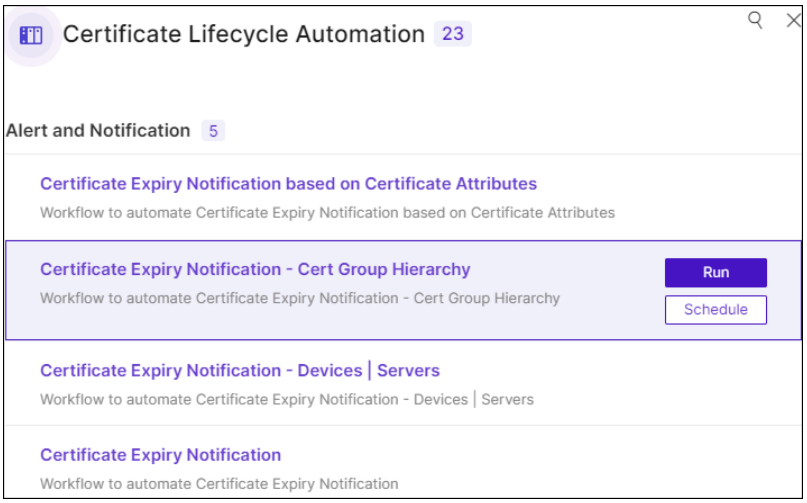
1. On the Request :: Overview page, from the menu on the left, click **View/Run**.
   The **Request :: View/Run** page is displayed.



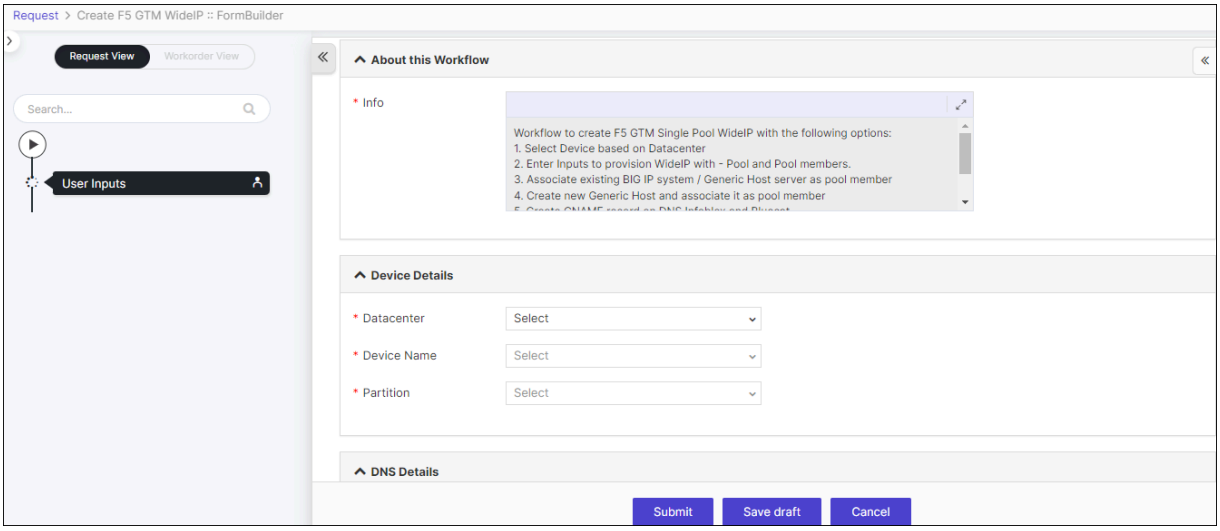
Note: Only enabled workflows are displayed on the **Request :: View/Run** page.

2. To trigger a workflow, on the **Request :: View/Run** page, search for the workflow.
3. Locate the workflow in the catalog and click the catalog name.
4. Hover your mouse over the workflow name, and then click Run **(Run)** button to initiate the workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The workflow is executed with the user inputs requested at the first stage.



Scheduling a Workflow

This task allows you to schedule a workflow to trigger once or daily, weekly, monthly and yearly as per requirement.

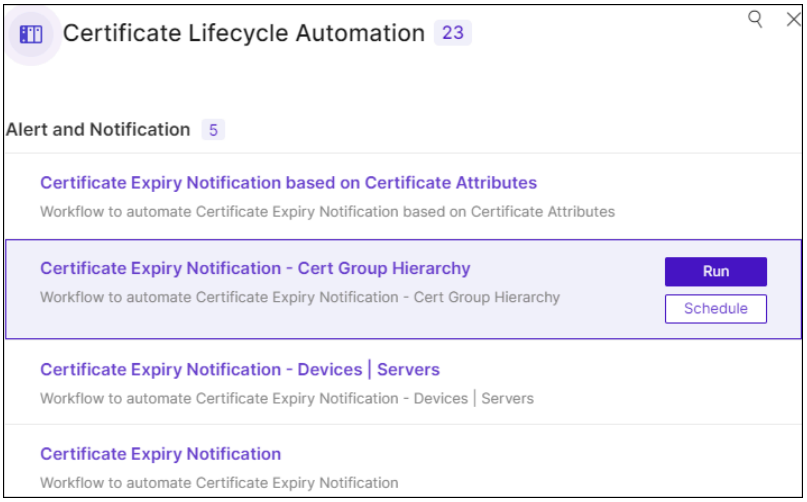
1. On the Request :: Overview page, from the menu on the left, click **View/Run**.
   The **Request :: View/Run** page is displayed.

   Note: Only enabled workflows are displayed on the **Request :: View/Run** page.

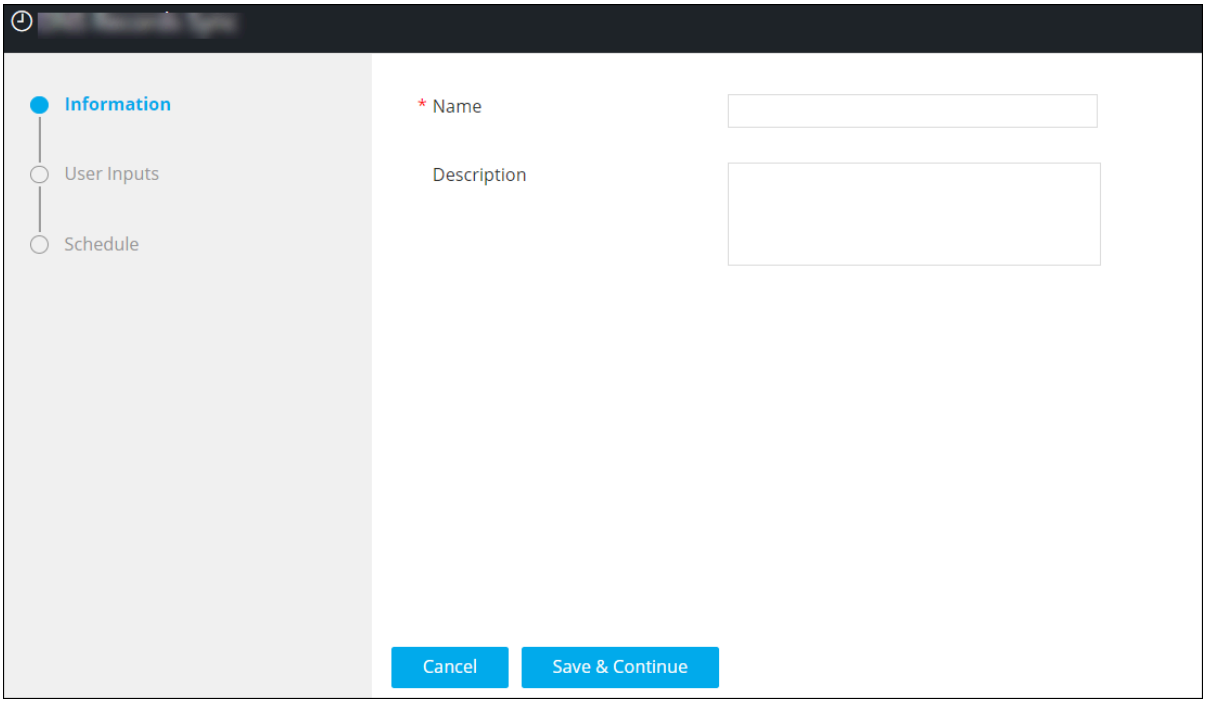
2. To schedule a workflow, on the **Request :: View/Run** page, search for the workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Locate the workflow in the catalog and click the catalog name.
4. Hover your mouse over the workflow name, and then click Schedule **(Schedule)** button.



The schedule workflow page is displayed.

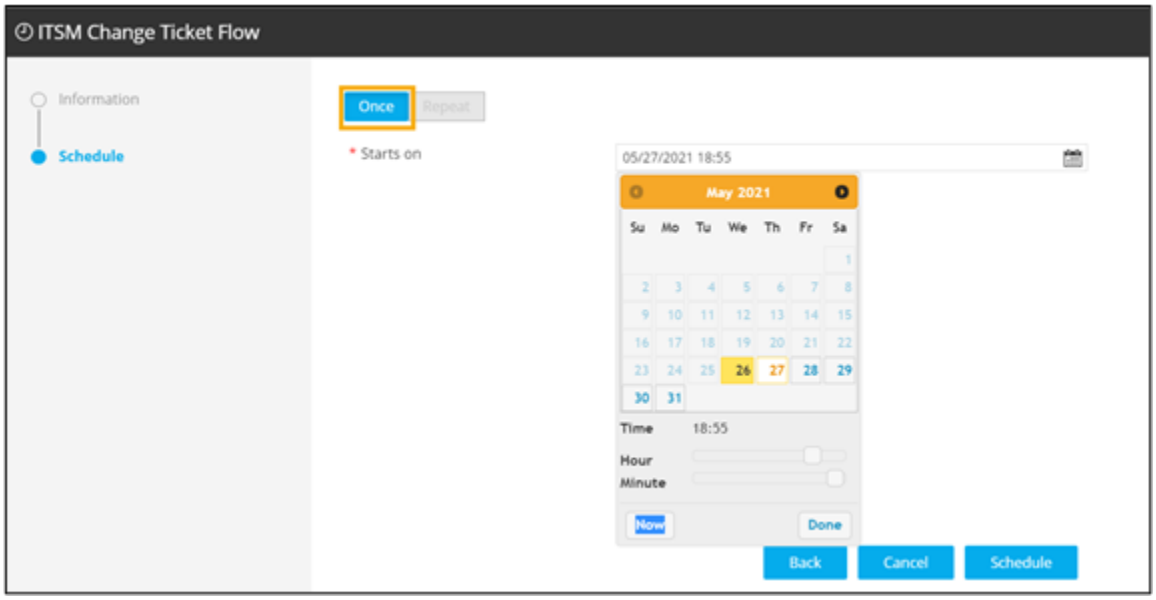


5. In the **Information** section, enter the **Name** of the scheduled job.
6. In the **User Inputs** section, enter the required field information.

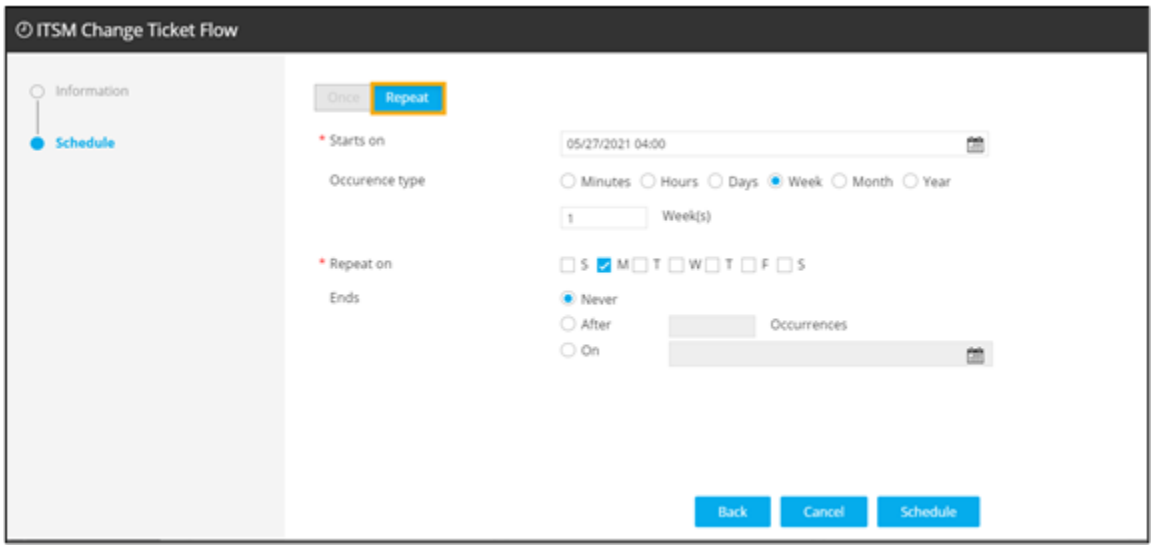
> **Note:** This field is displayed only if the workflow requires user inputs. The user inputs required will vary according to the workflow.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. In the **Schedule** section, assign the time for which the workflow is to be scheduled.
   - **Once**



   - **Repeat** - Provision to schedule the workflow to trigger daily, weekly, monthly and yearly.



8. Click **Schedule**.

   The scheduled workflow is added to the **Scheduled jobs** section.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Editing the Workflow Catalog Layout

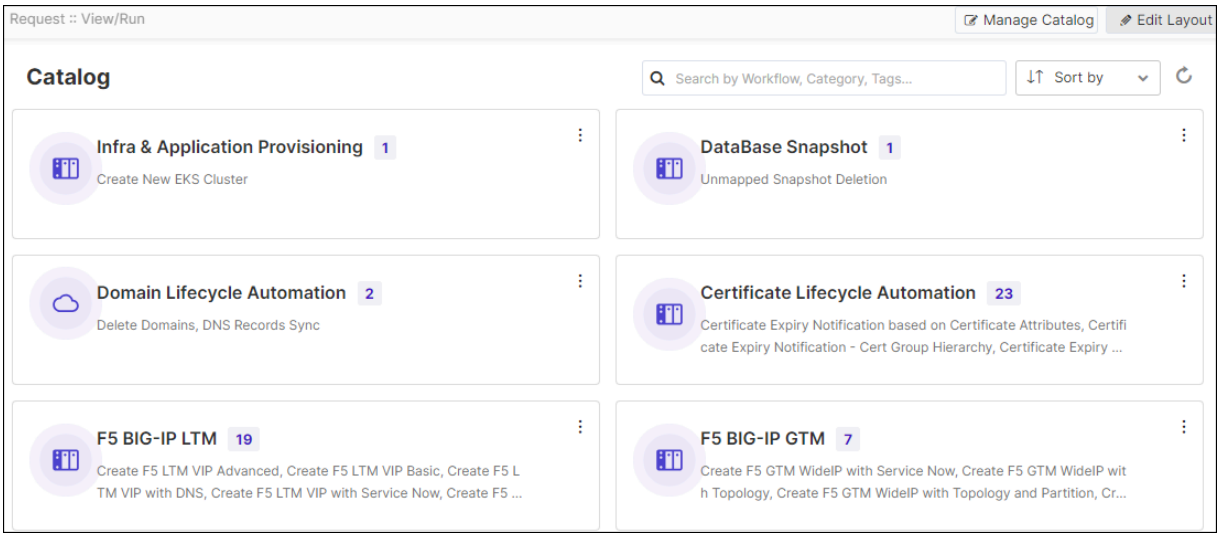
You can move the catalogs around and place them according to your preference on the View/Run page. You can also expand or collapse all workflow catalogs. Expanding the catalogs displays the list of workflows with a catalog.

To edit the layout:

1. On the Request :: Overview page, from the menu on the left, click **View/Run**.

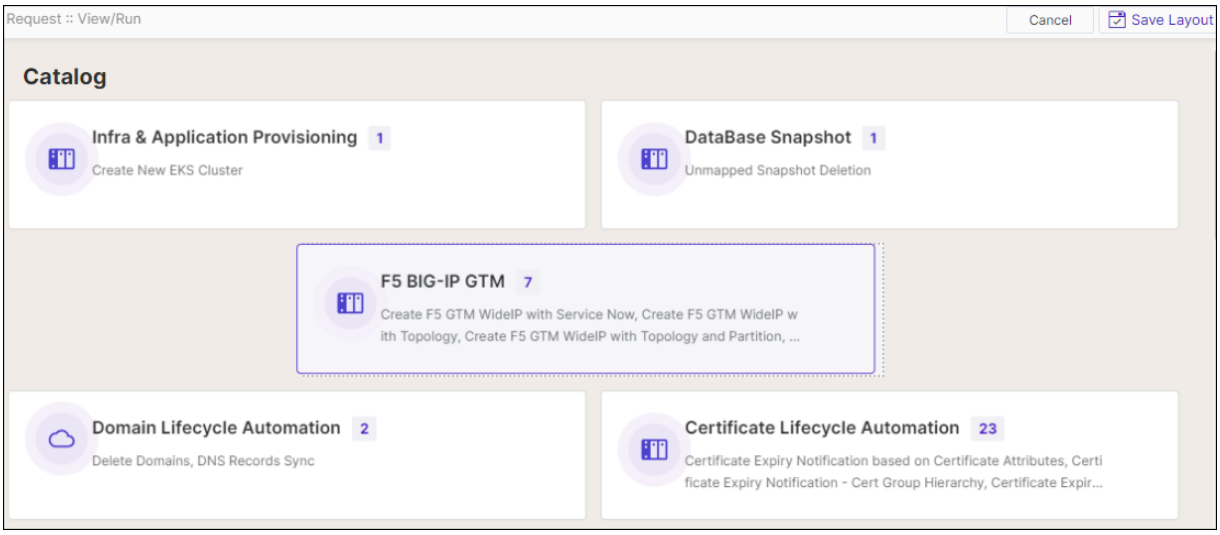
> **Note:** Only enabled workflows are displayed on the **Request :: View/Run** page.

2. On the **Request :: View/Run** page, from the top right corner of the screen, click **Edit Layout**.



3. Hold and drag the workflow catalog to another area on the page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. To save your layout preferences, from the top right corner of the page, click **Save Layout**.

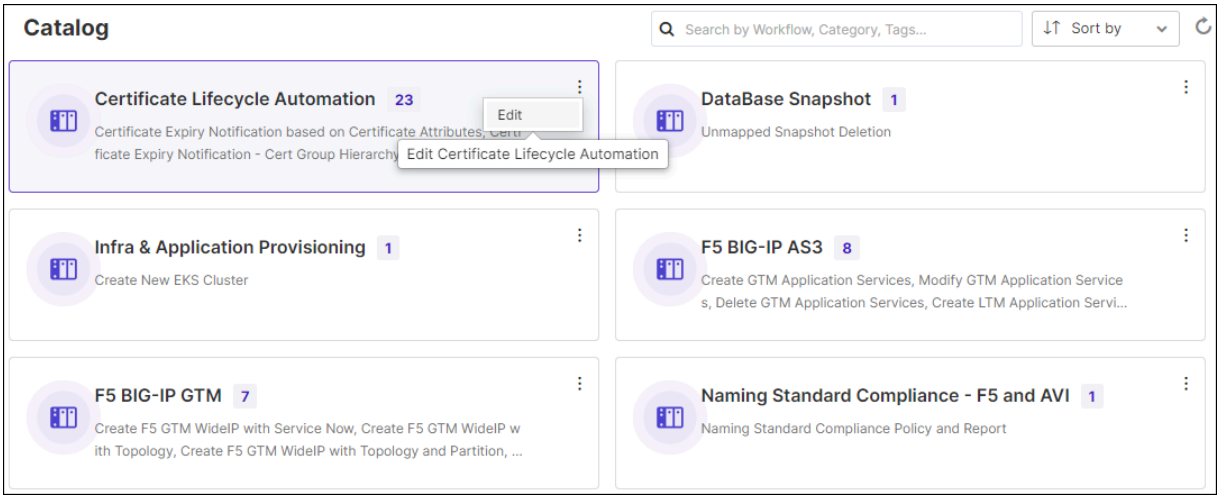
Editing Subcategories and Workflows within a Catalog

You can select the workflows to be placed under a workflow catalog and modify the order of sub-categories within the workflow catalog.

1. On the Request :: Overview page, from the menu on the left, click **View/Run**.

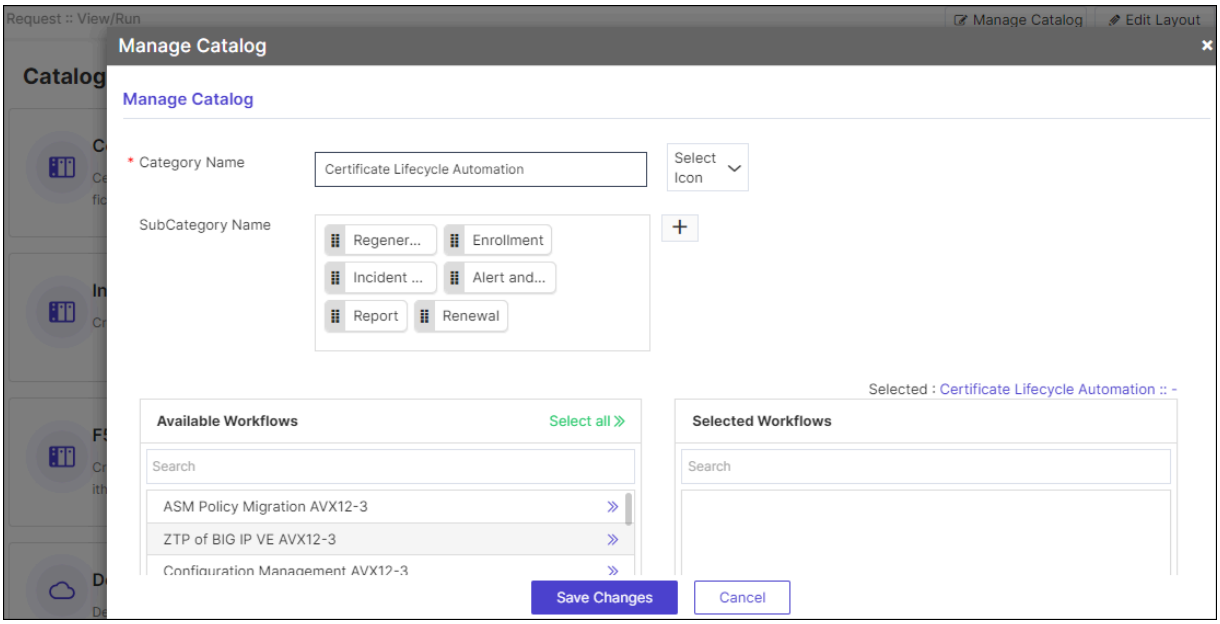
> **Note:** Only enabled workflows are displayed on the **Request :: View/Run** page.

2. On the **Request :: View/Run** page, from the top right corner of the screen, click ⋮.
3. From the options displayed, select **Edit**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. In the **Manage Catalog** window that is displayed, under **SubCategory Name**, hold and drag folders to modify the order in which they are displayed on the workflow card.

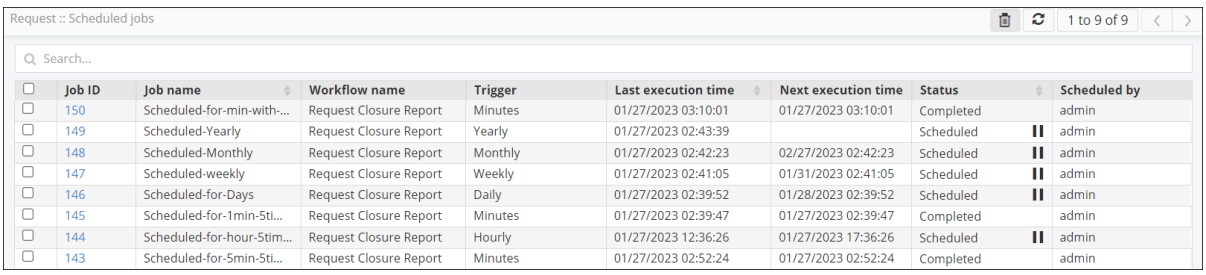


5. To add more workflows under a subcategory, select the subcategory and click » next to the workflow name under **Available Workflows**.
6. To move a workflow out of the selected workflows, click « next to the workflow under **Selected Workflows**.
7. Click **Save Changes**.

Scheduled Jobs

This section shows the inventory of scheduled workflows.

- Provision to ‘pause’ and ‘play’ a workflow.
- Displays details of execution time, trigger time and logs.
- Provision to ‘delete’ scheduled workflows.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. On the Request :: Overview page, from the menu on the left, click **Scheduled Jobs**.
2. To pause a scheduled job, on the **Request :: Scheduled jobs** page, click **(Pause)** icon.
3. To resume a paused scheduled job, click **(Resume)** icon.
4. To delete scheduled jobs, select the **Job ID**.
5. Click **(Delete)** icon.
6. In the **Confirmation** pop-up window, click **Yes**.
   The selected scheduled jobs are deleted.

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.
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 **(Clone)** icon from the command bar in the top right corner of the screen.
4. To refresh the **Request :: All** page, from the command bar, click **(Refresh)** icon.

Request :: Open

You can see a list of all the open requests on the **Request :: Open** page.

1. On the Request :: Overview page, from the navigation pane on the left, click **Open**.
   The **Request :: Open** page is displayed.
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 **(Clone)** icon from the command bar in the top right corner of the screen.
4. To refresh the **Request :: Open** page, from the command bar, click **(Refresh)** icon.

Request :: Closed

You can see a list of all the closed or completed requests on the **Request :: Closed** page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. On the Request :: Overview page, from the navigation pane on the left, click **Closed**.
   The **Request :: Closed** page is displayed.
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 **(Clone)** icon from the command bar in the top right corner of the screen.
4. To refresh the **Request :: Closed** page, from the command bar, click **(Refresh)** icon.

Request :: Failed

You can see a list of all the failed requests on the **Request :: Failed** page.

1. On the Request :: Overview page, from the navigation pane on the left, click **Failed**.
   The **Request :: Failed** page is displayed.
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 **(Clone)** icon from the command bar in the top right corner of the screen.
4. To refresh the **Request :: Failed** page, from the command bar, click **(Refresh)** icon.

Request :: Assigned

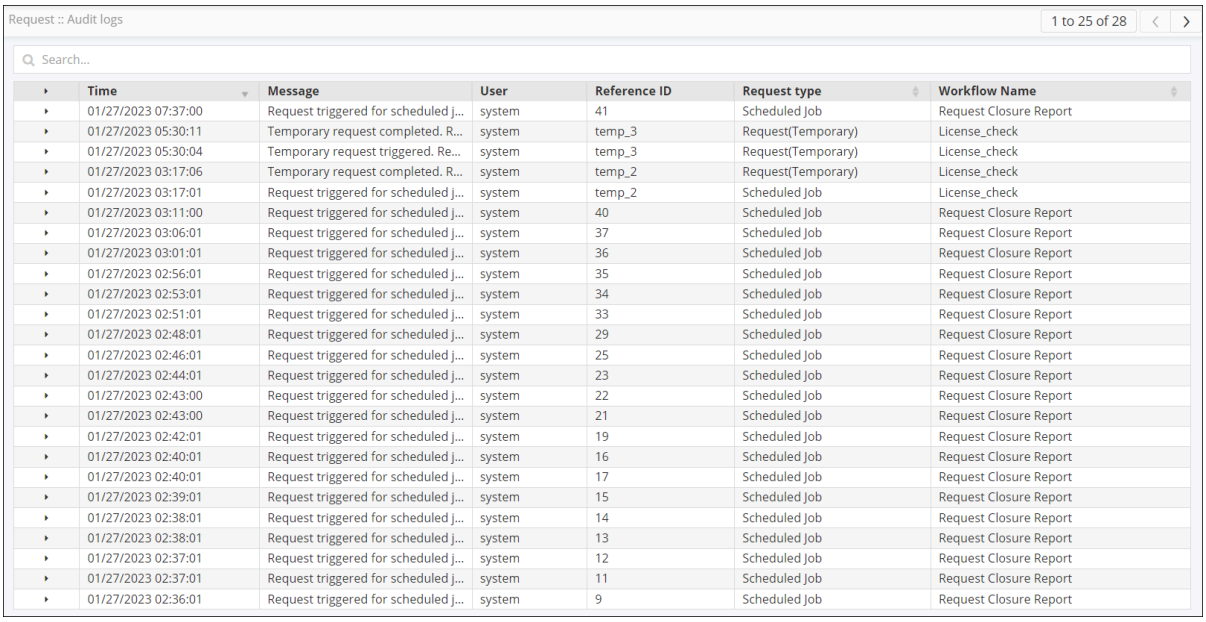
To see a list of all requests assigned to the logged in user:

1. On the Request :: Overview page, from the navigation pane on the left, click **Assigned**.
   The **Request :: Assigned** page is displayed.
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 **(Clone)** icon from the command bar in the top right corner of the screen.
4. To refresh the **Request :: Assigned** page, from the command bar, click **(Refresh)** icon.

Request :: Audit Logs

You can view the audit logs of all scheduled workflows and report-based workflows that were triggered on the **Request :: Audit Logs** page. Navigate to Request :: Overview > **Audit logs**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.


Request :: Audit logs

1 to 25 of 28

Search...

Time	Message	User	Reference ID	Request type	Workflow Name
01/27/2023 07:37:00	Request triggered for scheduled j...	system	41	Scheduled Job	Request Closure Report
01/27/2023 05:30:11	Temporary request completed. R...	system	temp_3	Request(Temporary)	License_check
01/27/2023 05:30:04	Temporary request triggered. Re...	system	temp_3	Request(Temporary)	License_check
01/27/2023 03:17:06	Temporary request completed. R...	system	temp_2	Request(Temporary)	License_check
01/27/2023 03:17:01	Request triggered for scheduled j...	system	temp_2	Scheduled Job	License_check
01/27/2023 03:11:00	Request triggered for scheduled j...	system	40	Scheduled Job	Request Closure Report
01/27/2023 03:06:01	Request triggered for scheduled j...	system	37	Scheduled Job	Request Closure Report
01/27/2023 03:01:01	Request triggered for scheduled j...	system	36	Scheduled Job	Request Closure Report
01/27/2023 02:56:01	Request triggered for scheduled j...	system	35	Scheduled Job	Request Closure Report
01/27/2023 02:53:01	Request triggered for scheduled j...	system	34	Scheduled Job	Request Closure Report
01/27/2023 02:51:01	Request triggered for scheduled j...	system	33	Scheduled Job	Request Closure Report
01/27/2023 02:48:01	Request triggered for scheduled j...	system	29	Scheduled Job	Request Closure Report
01/27/2023 02:46:01	Request triggered for scheduled j...	system	25	Scheduled Job	Request Closure Report
01/27/2023 02:44:01	Request triggered for scheduled j...	system	23	Scheduled Job	Request Closure Report
01/27/2023 02:43:00	Request triggered for scheduled j...	system	22	Scheduled Job	Request Closure Report
01/27/2023 02:43:00	Request triggered for scheduled j...	system	21	Scheduled Job	Request Closure Report
01/27/2023 02:42:01	Request triggered for scheduled j...	system	19	Scheduled Job	Request Closure Report
01/27/2023 02:40:01	Request triggered for scheduled j...	system	16	Scheduled Job	Request Closure Report
01/27/2023 02:40:01	Request triggered for scheduled j...	system	17	Scheduled Job	Request Closure Report
01/27/2023 02:39:01	Request triggered for scheduled j...	system	15	Scheduled Job	Request Closure Report
01/27/2023 02:38:01	Request triggered for scheduled j...	system	14	Scheduled Job	Request Closure Report
01/27/2023 02:38:01	Request triggered for scheduled j...	system	13	Scheduled Job	Request Closure Report
01/27/2023 02:37:01	Request triggered for scheduled j...	system	12	Scheduled Job	Request Closure Report
01/27/2023 02:37:01	Request triggered for scheduled j...	system	11	Scheduled Job	Request Closure Report
01/27/2023 02:36:01	Request triggered for scheduled j...	system	9	Scheduled Job	Request Closure Report

Configuring RBAC for Archive and Restore Requests

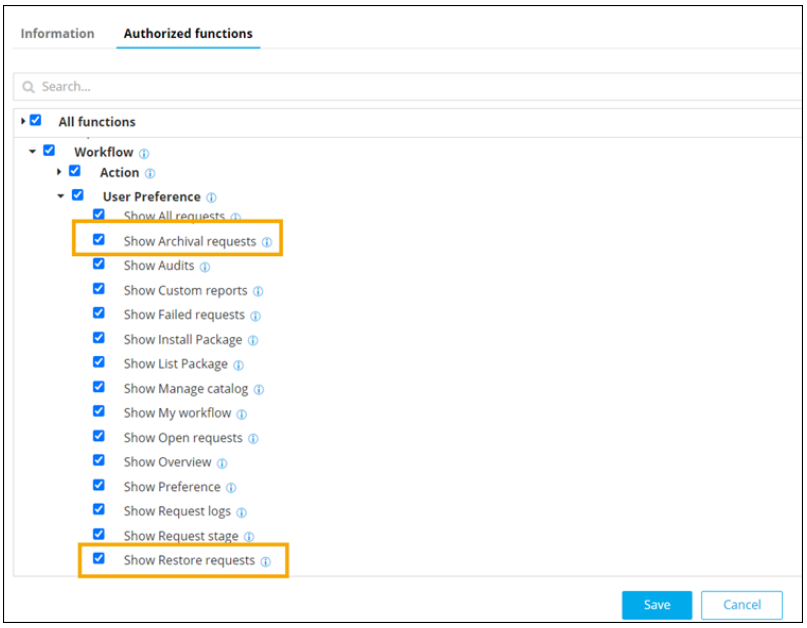
You can allow users to access the **Archive & Restore** section or restrict them from viewing this section on the Workflow Request page.

1. Go to the **Role** page in the Platform module.
2. Select the role for which you want to enable access. For example, **admin**.
3. Under **Authorized functions**, expand **Automation** > **Service Requests** > **User Preference**.

> **Note:** If you are using the old menu, under the **Authorized Functions** tab, expand **Request** > **Workflow**.

4. Under **User Preference**, select from the following checkboxes for the option to be displayed in the Service Requests console.
   - **Show Archival requests**
   - **Show Restore requests**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: Only the option that is selected under **User Preference** will be displayed under **Service Requests**.

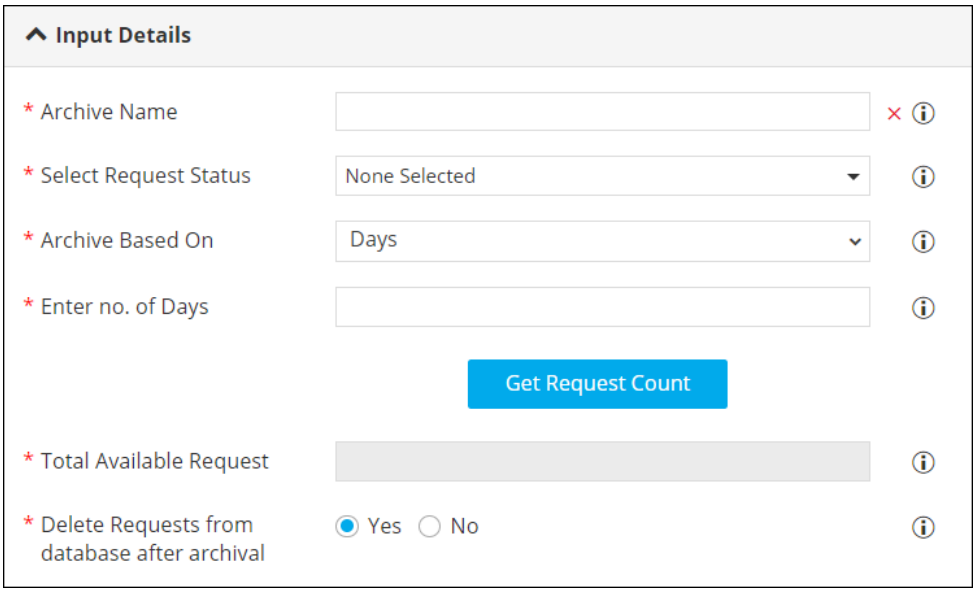
Archive Requests

You can archive workflow service requests by status, time period and so on. You can also schedule archivals and also purge service requests.

To trigger an archive workflow request:

1. On the Request :: Overview page, from the navigation pane on the left, click **Archive Requests**. The **Archive Workflow Requests** workflow is triggered with the workflow inputs requested at the first stage.
2. In the **Input Details** section, enter or select the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Input Details

Field	Description
***Archive Name**	Enter the name of the archive.
***Select Requests Status**	Select one or more type(s) of requests to be archived from the options available in the dropdown.
***Archive Based On**	Select the duration for which the requests will be archived as: • **Days**: This option allows you to enter the number of days for archiving service requests. • **Between Time**: This option allows you to select the Start and End date for archiving service requests.
***Enter no. of Days**	Enter the number of days for which you want to archive the service requests. **Note:** This field is displayed when you select **Days** in the **Archive Based On** field.
***Select Start Date**	Select the date and time from when the service requests are to be archived. **Note:** This field is displayed when you select **Between Time** in the **Archive Based On** field.
***Select End Date**	Select the date and time till which the service requests are to be archived.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
	Note: This field is displayed when you select **Between Time** in the **Archive Based On** field.
** : Mandatory fields*	

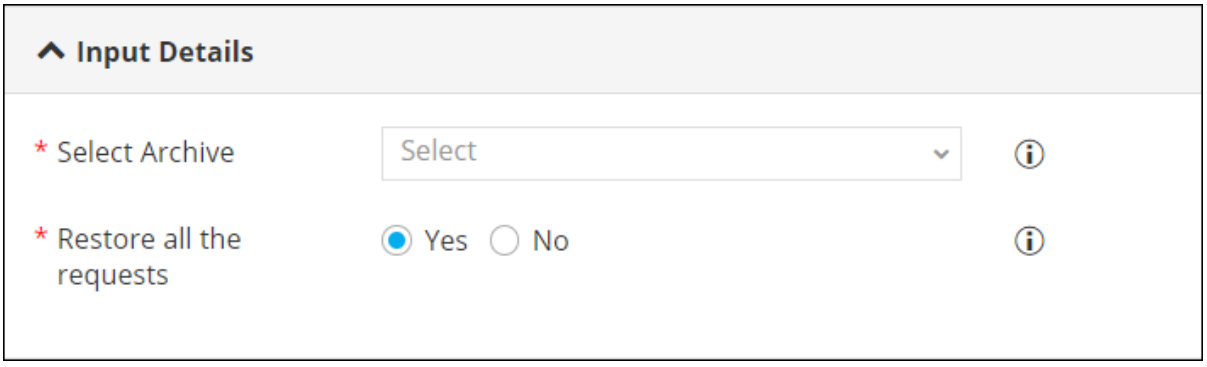
3. Click **Get Request Count**.
   The total number of requests that match the selection are displayed in the **Total Available Request** field.
4. To delete the requests from the database after archival, select **Yes**.
5. Click **Submit**.
   The service requests will be archived based on the selection.

Restore 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:

1. On the Request :: Overview page, from the navigation pane on the left, click **Restore Archives**.
   The **Restore Archived Requests** workflow is triggered with the workflow inputs requested at the first stage.



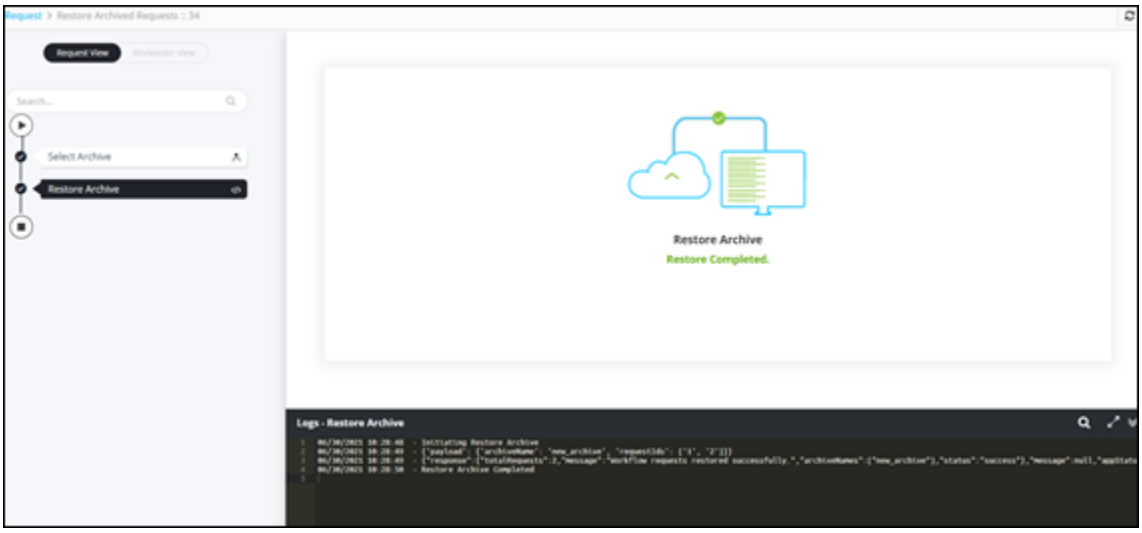
2. Under **Input Details**, select the archive that is to be restored.
3. To restore all the requests from the archive, select **Yes**.
4. To select specific request(s) from the archive, select **No**.
5. To auto populate the list of workflow requests, click **Get Request Details**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: This button is displayed only when you select **No** in the **Restore all the requests** field.

6. Select the workflows that you want to restore.
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.



Preferences

This section describes the following settings that can be configured as per your requirement in the Service Requests console:

- Selecting the Landing Page
- Viewing the View/Run page in Classic mode
- Managing the Catalogs

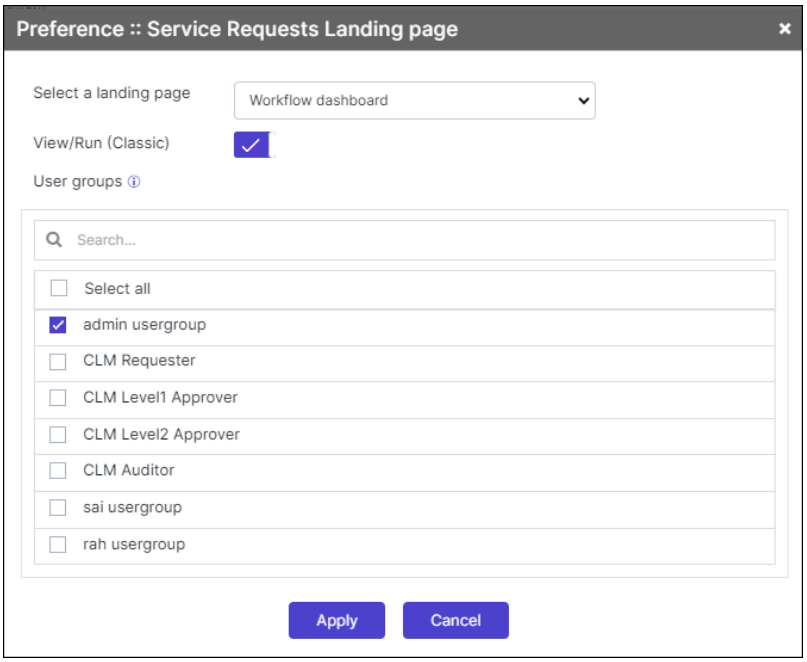
Selecting the Landing Page

You can select the page that is displayed when you access the Service Requests console.

1. On the Request :: Overview page, from the navigation pane on the left, click **Preference**.

   The **Preference :: Service Request Landing page** window is displayed.
2. In the **Preference :: Service Request Landing page** window, from the **Select a landing page** dropdown field, select a page that would display as a landing page when you login to the application.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



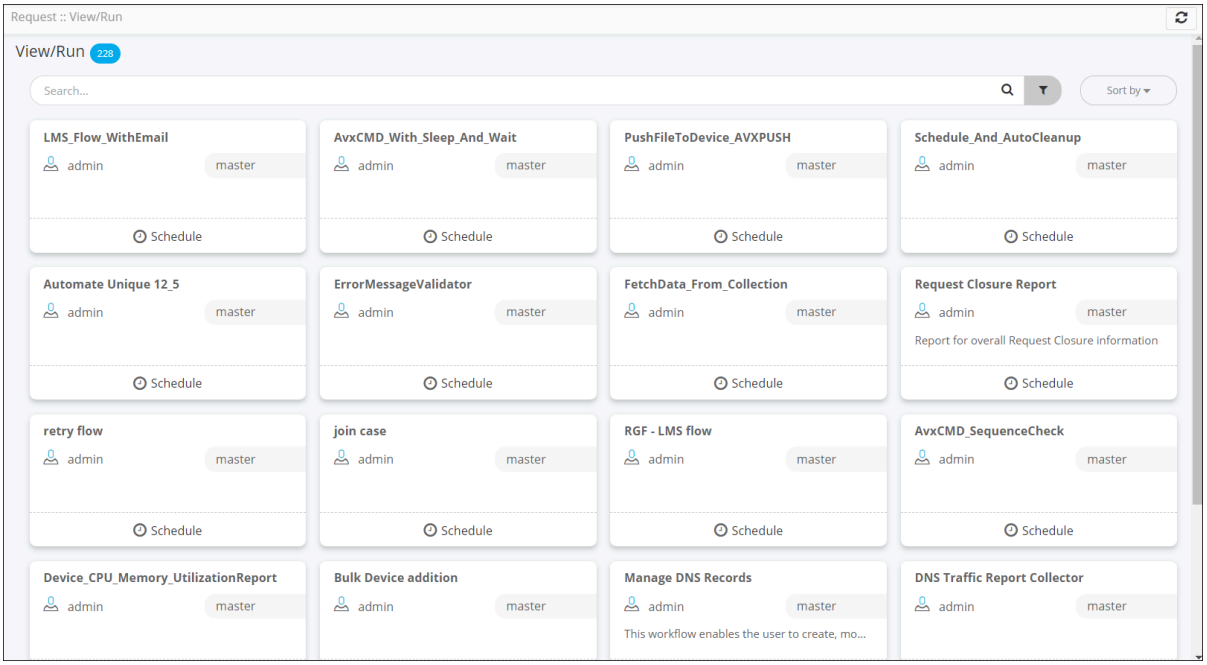
3. Turn On the toggle to **View/Run** page in (Classic) mode.
4. Under **User groups**, select the user groups for which the selected page will be set as the landing page.
5. Click **Apply**.

Viewing the View/Run page in Classic mode

1. On the Request :: Overview page, from the navigation pane on the left, click **Preference**.
   The **Preference :: Landing page** window is displayed.
2. To display the **View/Run** page in **Classic** mode, in the **Preference :: Landing page** window, turn on the **View/Run (Classic)** toggle.
3. Click **Apply**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The **View/Run** page is displayed in Classic mode.

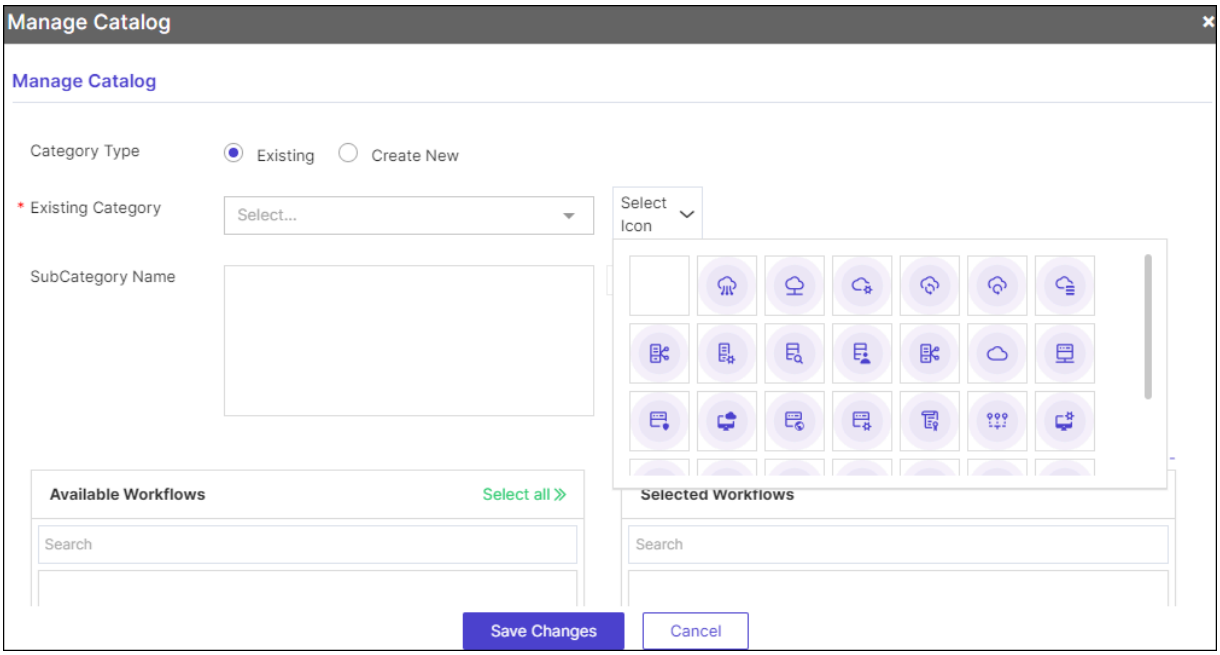


Managing the Catalogs

You can edit the catalog properties only 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 top right corner of the page, click **Manage Catalog**.
   The **Manage Catalog** window is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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.

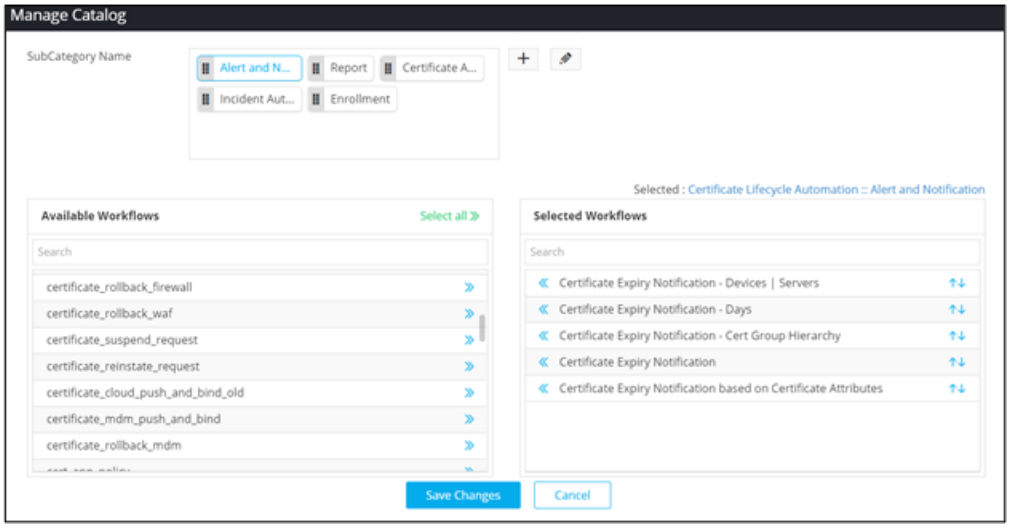
Options and Descriptions for Manage Catalogs

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 © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
+	Allows you to add more subcategories of workflows.
** : Mandatory fields*	

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**.

Service Request Actions

- Configuring RBAC for Activity logs
- Viewing the Workflow Log Summary
- Pausing/Resuming a Workflow
- Aborting a Workflow
- Rollback a Workflow

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

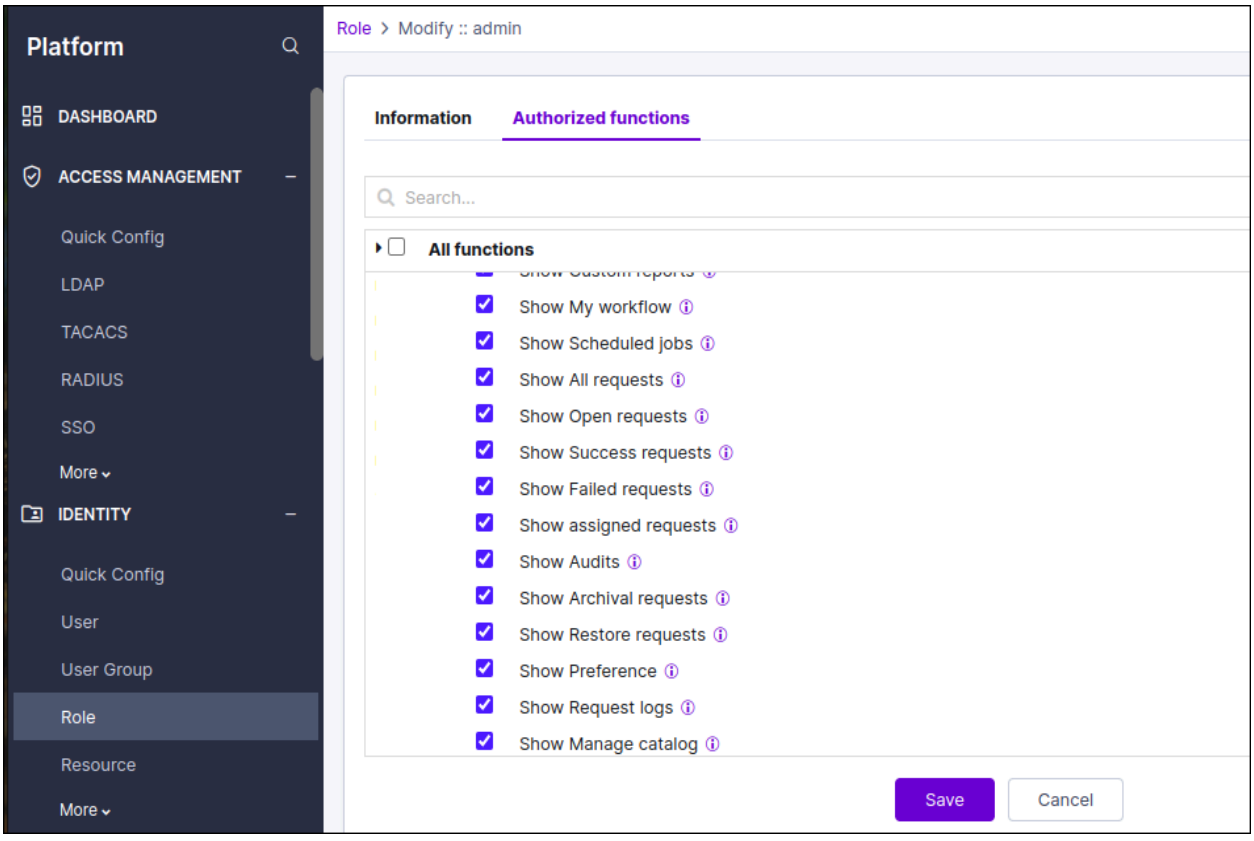
Configuring RBAC for Activity logs

The RBAC (Role-Based Access Control) includes the '**Show Request Log**' option to restrict access to logs in visual workflow requests.

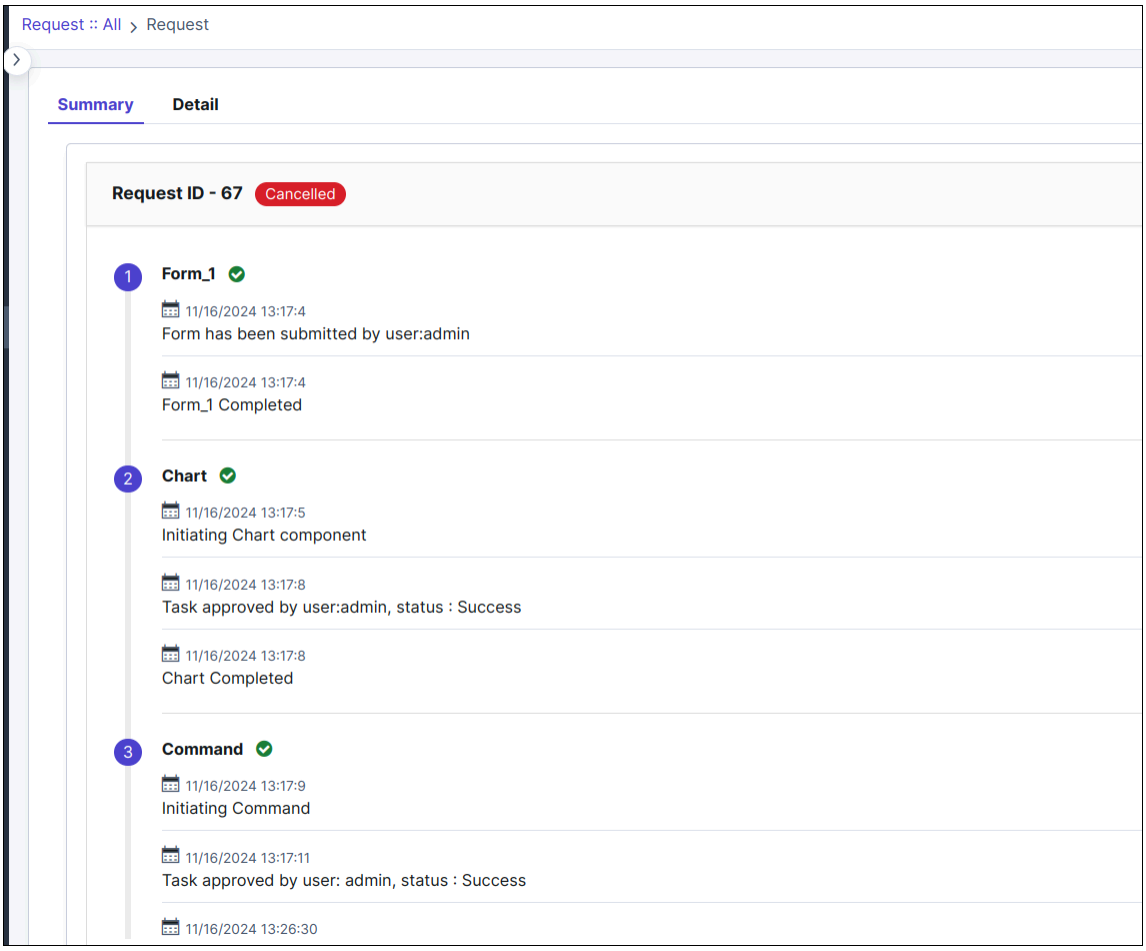
- By default, this permission is enabled for all existing roles.
- This feature applies exclusively to Service Tasks (UI tasks remain unchanged).
- In the activity log for Service Tasks, only 'Task initiated' and 'Task completed' entries will be displayed.
- Additionally, the logs for Service Tasks will be hidden when a request is executed.

1. Go to **Menu** > **Platform** module > under **IDENTITY**, select **Role**.
2. Select the role for which you want to enable access. For example, **admin**.
3. Under **Authorized functions**, expand **Automation** > **Service Requests** > **User Preference**.
4. Under **User Preference**, ensure that the **Show Request Log** checkbox is selected/unchecked based on the access to provide.

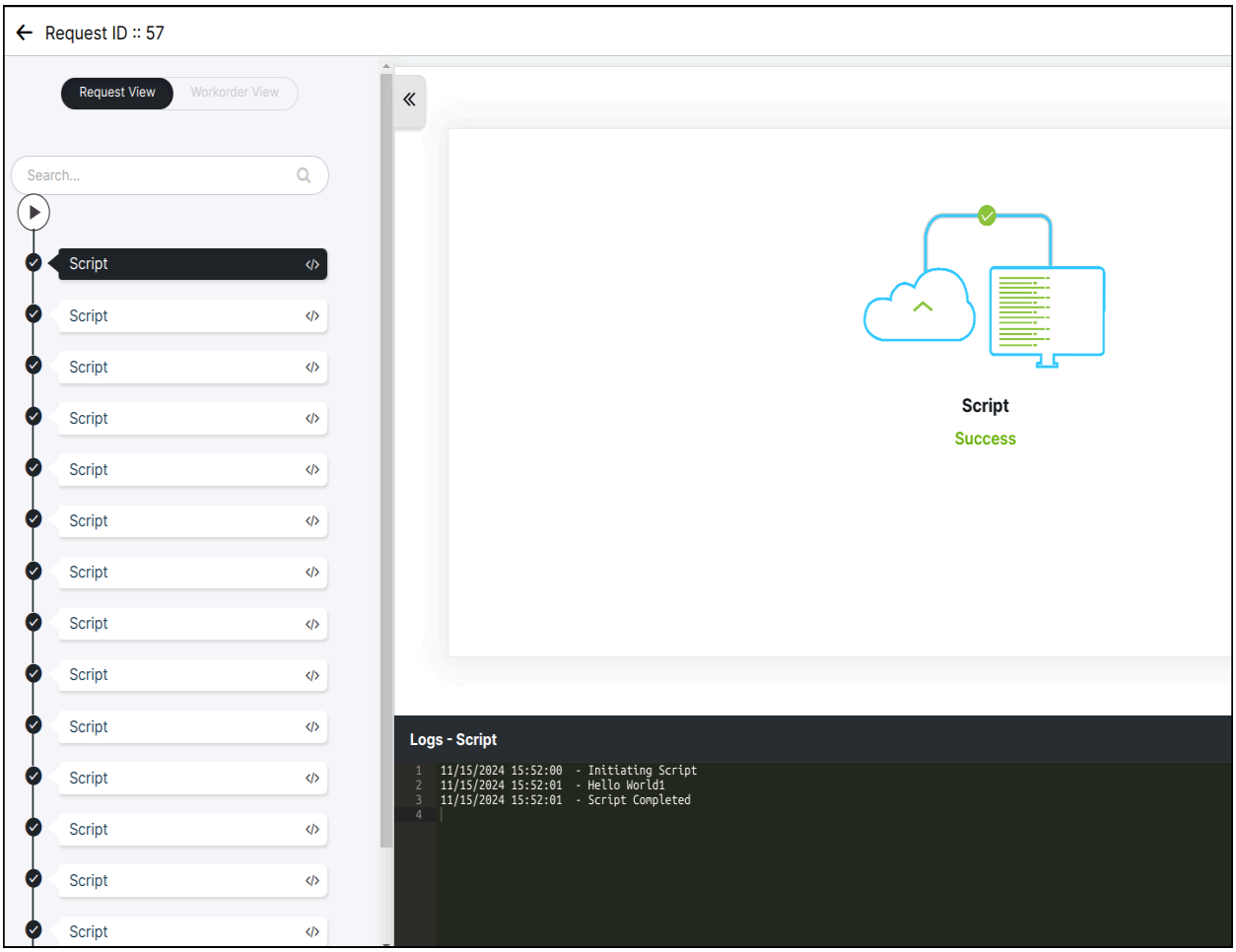
When ACF (**Show Request Log**) is **enabled**, the logs can be viewed in the Activity Log and Script execution task as shown below.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

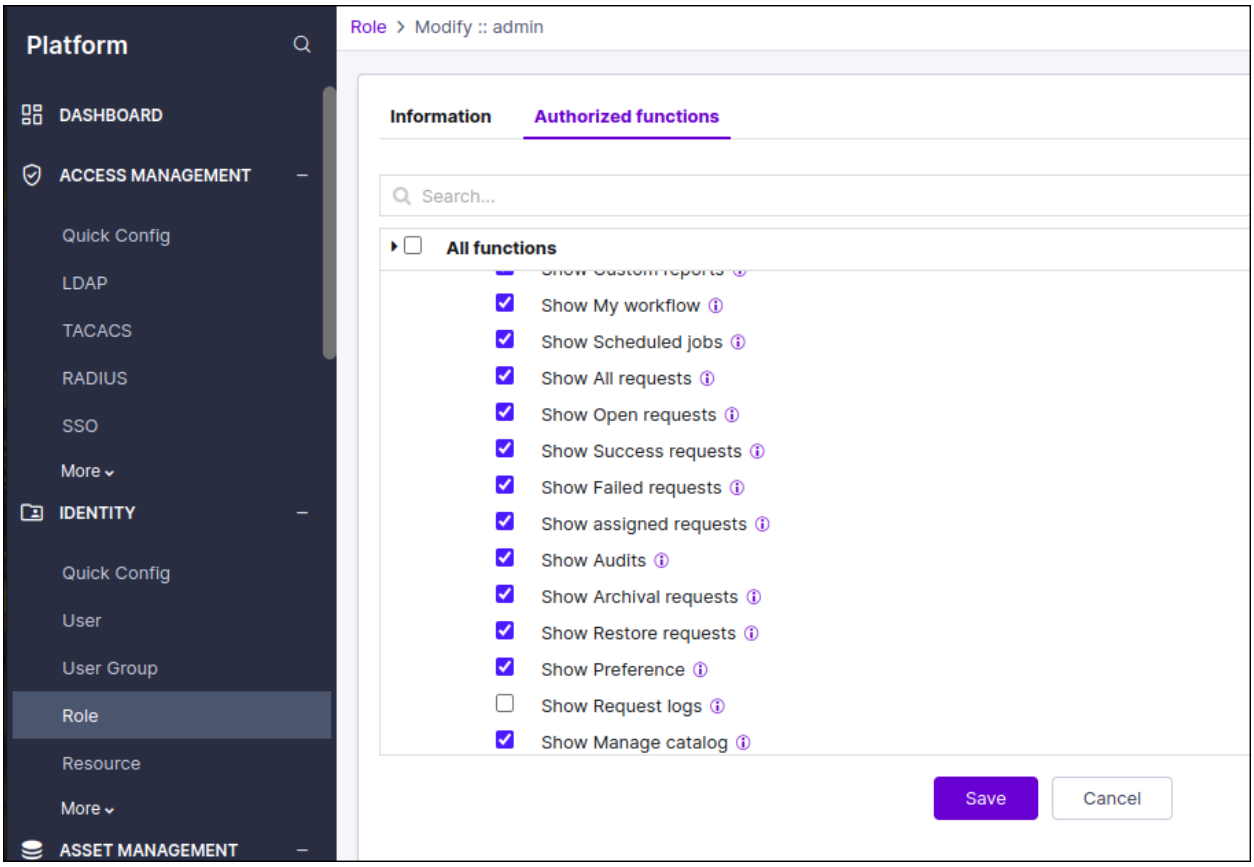


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

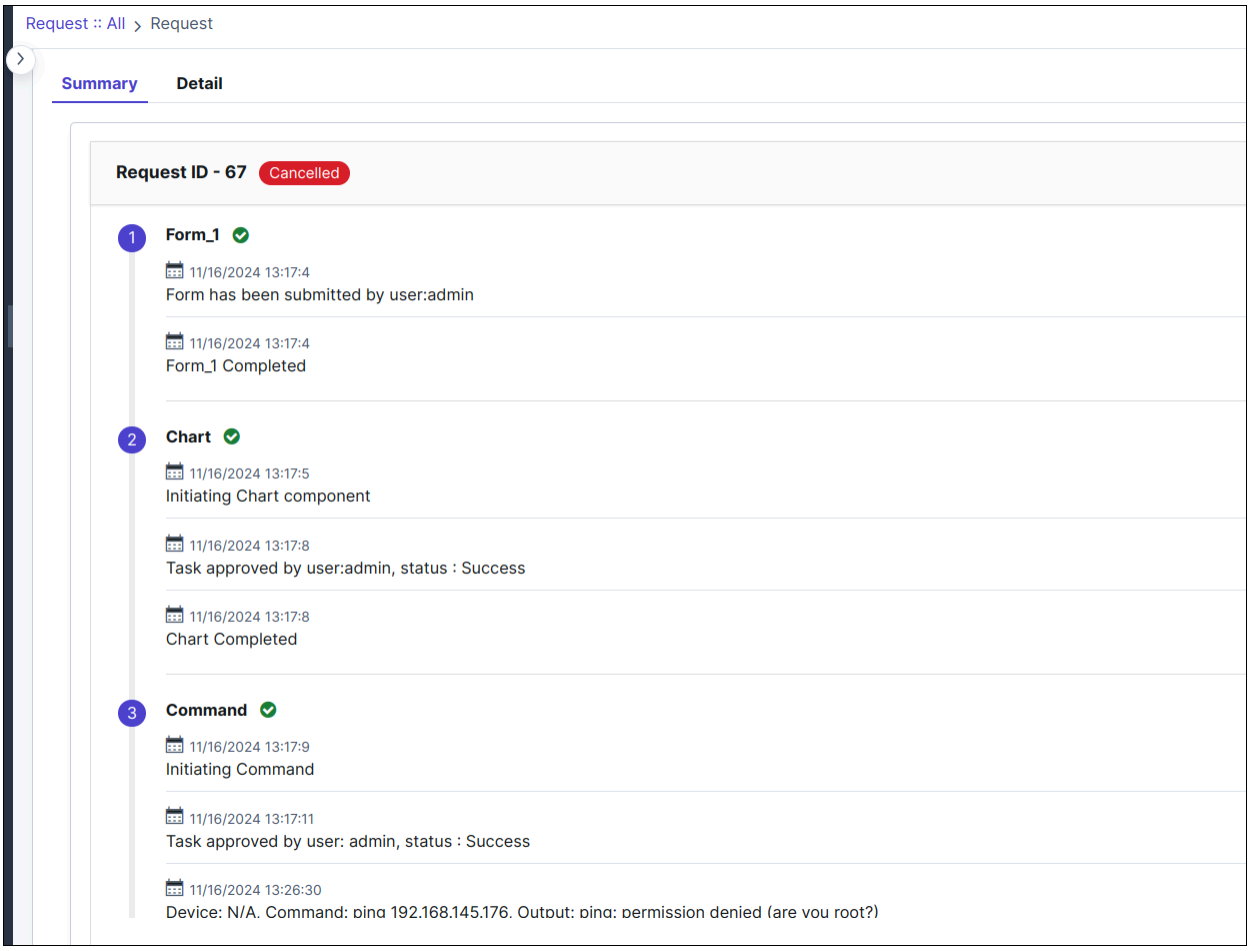


When ACF (**Show Request Log**) is **disabled**, only the 'initiate' and 'complete' logs are displayed in the Activity Log as shown below. This is only applicable for **Service Tasks**.

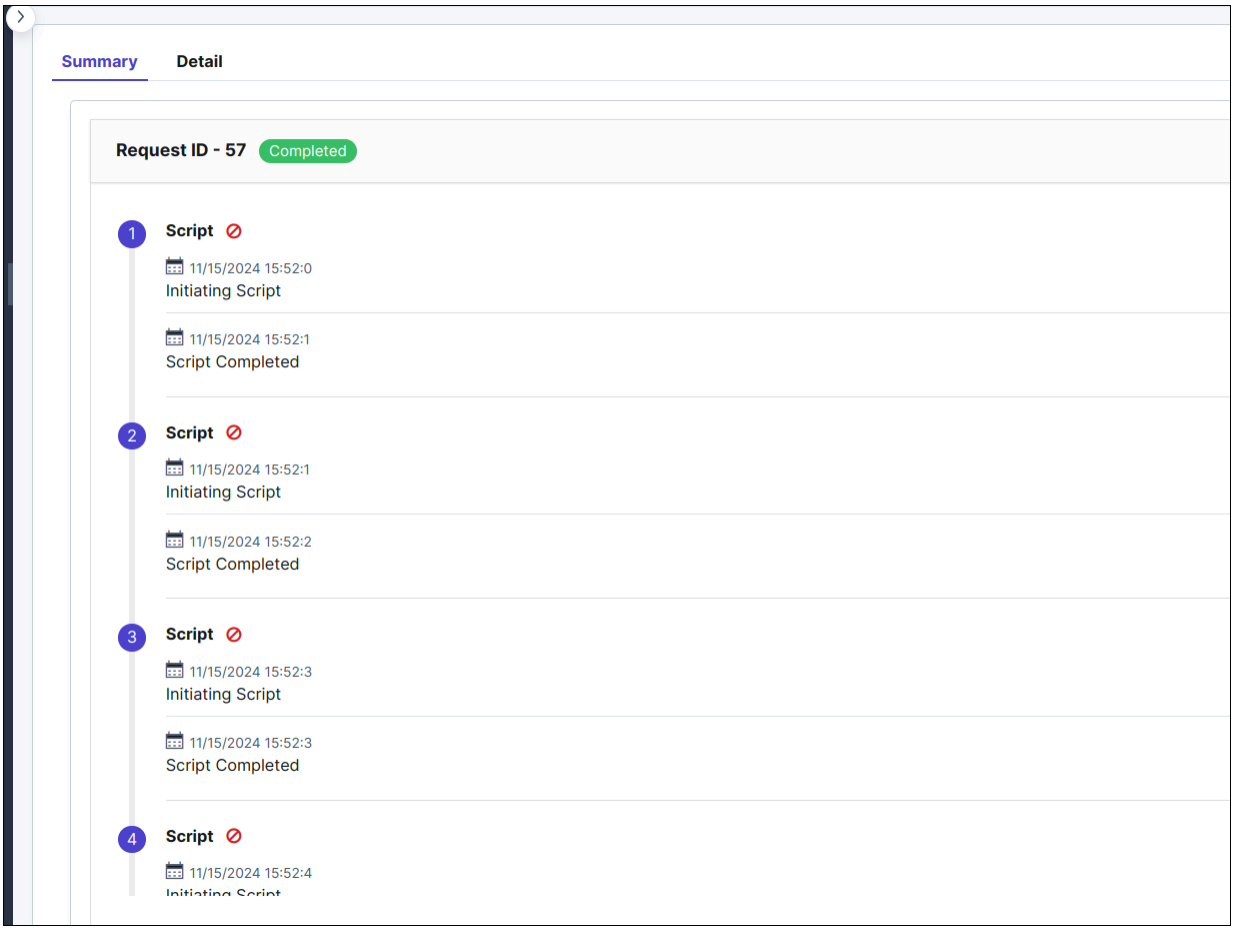
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.



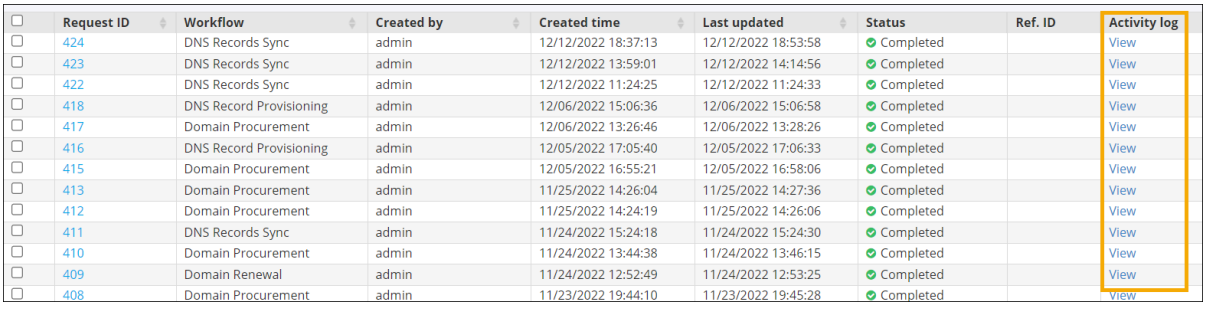
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Viewing the Workflow Log Summary

To view the activity log of a specific workflow:

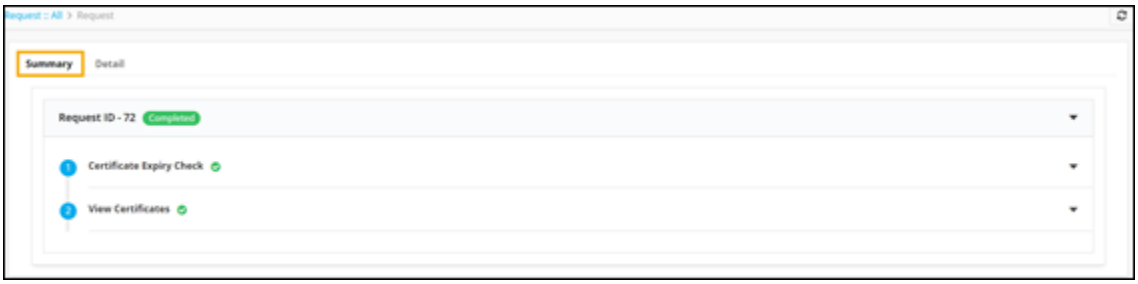
1. On the Request :: All page, under the **Activity log** column, click **View** next to the workflow for which you want the summary.



Request ID	Workflow	Created by	Created time	Last updated	Status	Ref. ID	Activity log
424	DNS Records Sync	admin	12/12/2022 18:37:13	12/12/2022 18:53:58	Completed		View
423	DNS Records Sync	admin	12/12/2022 13:59:01	12/12/2022 14:14:56	Completed		View
422	DNS Records Sync	admin	12/12/2022 11:24:25	12/12/2022 11:24:33	Completed		View
418	DNS Record Provisioning	admin	12/06/2022 15:06:36	12/06/2022 15:06:58	Completed		View
417	Domain Procurement	admin	12/06/2022 13:26:46	12/06/2022 13:28:26	Completed		View
416	DNS Record Provisioning	admin	12/05/2022 17:05:40	12/05/2022 17:06:33	Completed		View
415	Domain Procurement	admin	12/05/2022 16:55:21	12/05/2022 16:58:06	Completed		View
413	Domain Procurement	admin	11/25/2022 14:26:04	11/25/2022 14:27:36	Completed		View
412	Domain Procurement	admin	11/25/2022 14:24:19	11/25/2022 14:26:06	Completed		View
411	DNS Records Sync	admin	11/24/2022 15:24:18	11/24/2022 15:24:30	Completed		View
410	Domain Procurement	admin	11/24/2022 13:44:38	11/24/2022 13:46:15	Completed		View
409	Domain Renewal	admin	11/24/2022 12:52:49	11/24/2022 12:53:25	Completed		View
408	Domain Procurement	admin	11/23/2022 19:44:10	11/23/2022 19:45:28	Completed		View

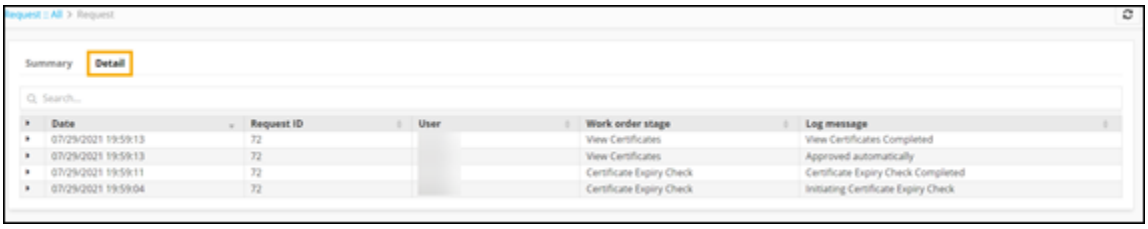
A **Summary** of the workflow stages is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: The RBAC (Role-Based Access Control) includes the '**Show Request Log**' option to restrict access to logs in visual workflow requests. Refer to the section Configuring RBAC for Activity logs.

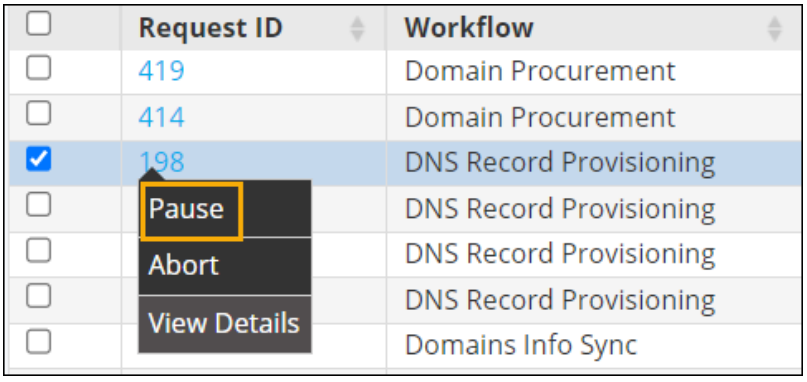
2. To view the workflow log requests details, click **Detail**.
   A detailed log of the workflow stages is displayed.



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 on the **Request ID** of the workflow that you want to pause.
2. From the options displayed, select **Pause**.



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

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

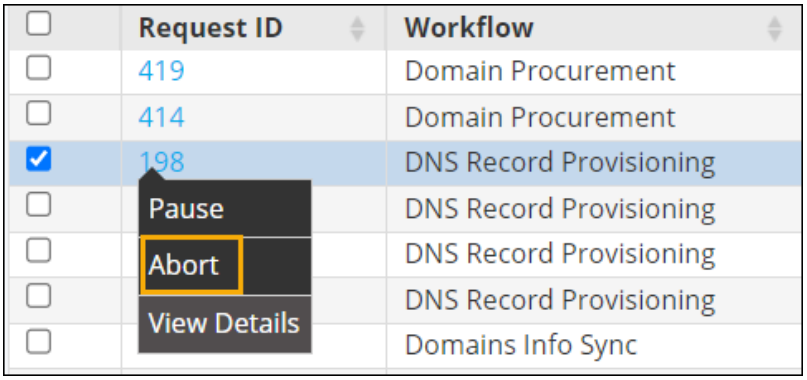
4. To resume the workflow, right-click and select **Resume**.
5. Click **Yes** in the confirmation pop-up window.

   The workflow resumes.

Aborting a Workflow

You can only abort workflows that are still open or in-progress.

1. On the Request :: All page, right-click on the **Request ID** of the workflow that you want to abort.
2. Select **Abort** from the available options.



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

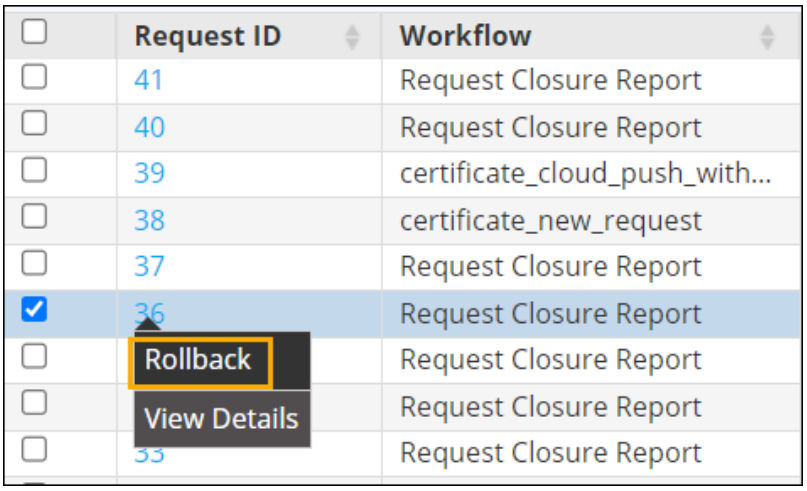
> **Note:** You can abort a workflow only if it is in the **In progress** state.

Rollback a Workflow

You can manually rollback a workflow request that has been completed (either successfully or failed).

1. On the Request :: All page, right-click on the **Request ID** of the workflow that you want to rollback.
2. From the options displayed, select **Rollback**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



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

> **Tip:** You can also rollback a workflow, by selecting the workflow and clicking in the upper right corner of the screen.

Rollback :: (Number) pop-up is displayed.

4. Select the **Rollback type** as "**Request**" or "**Workorder**".
   a. If the **Rollback type** is set to **Request**, click **Rollback**.
   b. If the **Rollback type** is set to **Workorder**, choose the **Work order ID** from the table, and then click **Rollback**.

Automation and Service Catalog

Once you have designed your workflows, you can view them all in one place through AppViewX's self-service catalog. To access the AppViewX Service Catalog, navigate to **Automation** > **SERVICE CATALOG**.

- 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: For more information, refer to the Service Catalog User Guide.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Chapter 2: Automation API Guide

AppViewX's Automation module is a hybrid cloud automation and orchestration platform that allows for simple and complex network automation within your infrastructure. The module 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. Its primary interface is through both the Web GUI interface and a HTTP Restful API. A developer would use this API for programmatic access.

This guide provides information about the AppViewX exposed APIs intended for use in **Visual Workflow** actions.

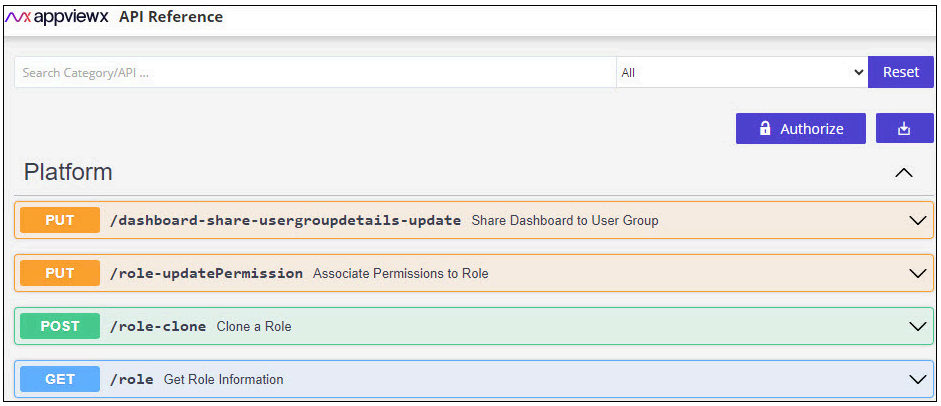
API Reference

This applies only to AppViewX customers. To try the AppViewX exposed APIs on Swagger, follow these steps:

1. From the top right corner of your screen, click {...} **(API Reference)** icon.

   > **Note:** The **API Reference** icon is displayed for the Admin user and users with ACF permission enabled for API Reference.

   You will be redirected to **AppViewX API Reference** page.



2. Click any API to **Try it out**.

Enable ACF Permission for API Reference

To enable the ACF permission for non-admin roles to access the API reference, follow these steps:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

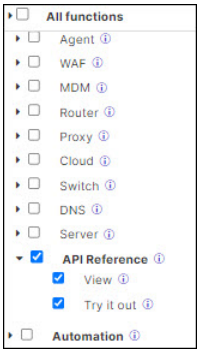
1. Go to **Platform** module **IDENTITY** > **Role**.

   You will be redirected to **Role** page.

2. Click on the role name to enable the ACF permission.

   You will be redirected to the **Modify :: [RoleName]** page, with the **Information** tab open by default.

3. Switch to the **Authorized Functions** tab and expand the **Platform** by clicking **(Expand)** icon.
4. To assign **API Reference**, select the checkbox beside **API Reference**.



5. Click **Save**.

Best Practices for Working with the AppViewX API

- **Use appropriate HTTP methods**

  Ensure that the correct HTTP method is used for each operation (e.g., GET for retrieval, POST for creation).

- **Handle errors gracefully**

  Implement proper error handling in your application to manage API responses.

- **Use secure storage**

  Store access tokens securely and avoid hardcoding them in your application code.

- **Implement pagination**

  For endpoints that return large datasets, implement pagination using limit and offset parameters.

- Understanding the AppViewX Automation API
- Authentication Using a User Account
- Authentication Using a Service Account
- Submit Request

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• Update Request
• Get Current Task Detail of the Request
• Get Request Summary/Log
• Get Associated Workorder
• Get Tasks which are in InProgress State
• Upload File
• Download File
• Delete File
• Abort Request
• Clone Request
• Get All Requests
• Pause Request
• Resume Request
• Generate Commands
• Generate Commands And Implement
• Rollback Request
• Execute Hook
• Collection Operations
• Get Collection Details
• Collection Exists in DB
• Update Workflow Resource Permission

Understanding the AppViewX Automation API

The AppViewX API is a programmatic way to get data in and out of the AppViewX subsystems. With access to RESTful AppViewX APIs, you can leverage the raw potential of AppViewX. It provides a powerful way to channel the data into native business applications. This document comprises of module-wise APIs used in AppViewX.

RESTful HTTPS Requests

Type	Description
GET	GET requests, retrieve resource representation/information only and not to modify it.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Type	Description
POST	POST APIs create new subordinate resources. For example, a file is subordinate to a directory containing it or a row is subordinate to a database table. In terms of REST, POST methods are used to create a new resource into the collection of resources.
PUT	PUT APIs are used to update existing resources (if a resource does not exist then API may decide whether to create a new resource or not).
DELETE	DELETE APIs are used to delete resources (identified by the Request-URI).

Requests

Each endpoint URL is built in the same way by the following structure:

```
http://<IP/HostName/TenantName>:<GWPORT>/avxapi/<Endpoint>?<gwsource>
```

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Structure

All endpoints accept a request structure that should consist of JSON formatted data. To ensure the request is accepted, set the header **Content-Type: application/json**.

The following example shows a request to add a resource:

```
{
"payload": {
"name": "resource_1",
"description": "This is a sample resource."
}
}
```

Response Structure

The Content-Type of the response is typically determined by the Content-Type header, and for most endpoints, it will be application/json. All requests that reach the server, regardless of the response code, will retrieve a response body. A successful request will contain a body with the requested information, for example:

```
https://appviewxapi.com/avxapi/resource?gwsource=external
```

Returns the following JSON structure that a resource is added:

```
{
"response": "Resource added successfully",
"message": "Resource added successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Description of Server Responses

HTTP Code	Response Message
200 OK	The request was successful (some API calls may return 201 or 202 instead).
400 Bad Request	The request is not understood or required parameters are missing.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	Response Message
401 Unauthorized	Authentication failed or the user doesn't have permissions for the requested operation.
409 Forbidden	Access denied.
404 Not Found	Resource not found.
429 Too many requests	The number of requests to the service has crossed the threshold.
503 Service unavailable	The client cannot communicate with the service.
504 Gateway timeout	The given request has exceeded the expected time.

URI Scheme

- **Host :** {url}
- **BasePath :** /avxapi
- **Schemes :** HTTPS
- **URL :** https://{url}/avxapi

Types of Accounts in AppViewX

There are two types of accounts in AppViewX:

- **User Accounts**: These are used by actual users.
- **Service Accounts**: These are used by system services such as web servers, automation tools, and so on.

AppViewX recommends using a Service Account for accessing APIs from automation tools. Service Accounts are supported with oAuth standard for a more secure and standard way of accessing APIs.

Note: AppViewX supports both User Account and Service Account for accessing APIs.

Authentication Using a User Account

For accessing APIs, you can login via two types of accounts:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- User account

  A **User account** represents an individual person interacting with the application or the system. User accounts are used for accessing the system on behalf of a user.

  For accessing APIs with a user account, you need to get the session ID by providing a username and password in the login API. This session ID can then be used for accessing other APIs.

  Note: You can also use the username and password in all API calls instead of the sessionId. However, this is not recommended.

- Service account

  A **Service account** represents a non-human entity such as an application or a service. It is used for automated processes or system-to-system interactions without human intervention.

  For accessing APIs with a service account, you need to get the Access Token by providing Client ID and Client Secret in get-service-token API. This Access Token can then be used for accessing other APIs.

  Note: Access Token Validity is 30 minutes by default and it can be configured in **Settings** > **Authentication** > **oAuth Settings**.

- Retrieve session ID using login API
- Using Session ID for further API calls

Retrieve session ID using login API

This API used to retrieve the session ID using the login API for secure authentication and access to system resources.

Before you begin

- Make sure you have valid login credentials (Username and Password) for accessing the system.
- You cannot use OAuth credentials (Client ID and Client Secret) for login.
- To access the APIs using the service token, use the API with the Service Account.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Structure

Endpoint	`/login`
Type	`POST`
Sample URL	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/login?gwsource=external` To understand the elements of the sample URL, click here.
Headers	
Content-Type	application/json
Request timeout period	15 minutes

Input Parameters

Name	Description
username *Header*	(Mandatory) Use login name of the user. **Type:** String **Example:** "admin"
password *Header*	(Mandatory) Password for the username. **Type:** String **Example:** "AppViewX@123"
otp *Header*	(Mandatory only if MFA is enabled) If MFA is enabled, enter the OTP received on your registered email ID in the header. Multifactor authentication (MFA) is a security mechanism that requires users to provide two or more verification factors to gain access to a resource If MFA is enabled, and you try to login with only the username and password, you will get the following error upon execution of the API: **MFA is enabled. We have sent an OTP to your email ID: aaa********r@appviewx.com**. In this case, ensure that the OTP is included in the header and try logging in again.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Type: String **Example:** "OTP : **609700**"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Type:** String **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external **Type:** String

Response Structure

- **Status Code:** 200 Ok
- **Message:** Login Successful
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	The response contains the attributes needed to retrieve the session ID.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
status	Indicates the overall status of the response. The values can be: • SUCCESS • FAILURE
appStatusCode	An application-specific status code, if applicable.
statusDescription	Description of the status, if available.
sessionId	Unique identifier for the session.
lockDownPeriod	Number of login attempts remaining.
termsAccepted	
passwordExpiryMsg	
emailId	

Status Codes

HTTP Code	appStatusCode	Response Message
200 OK	NA	Login successful
400 Bad request	ACCT_AUTH_001	Username or password cannot be null or empty.
401 Unauthorized	ACC_AUTH_022	Login failed. Invalid credentials.
401 Unauthorized	ACC_AUTH_006	Login failed. Invalid credentials.

Sample Request/Response

Use Case

Login to the application with a username and password.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/login?gwsource=external
```

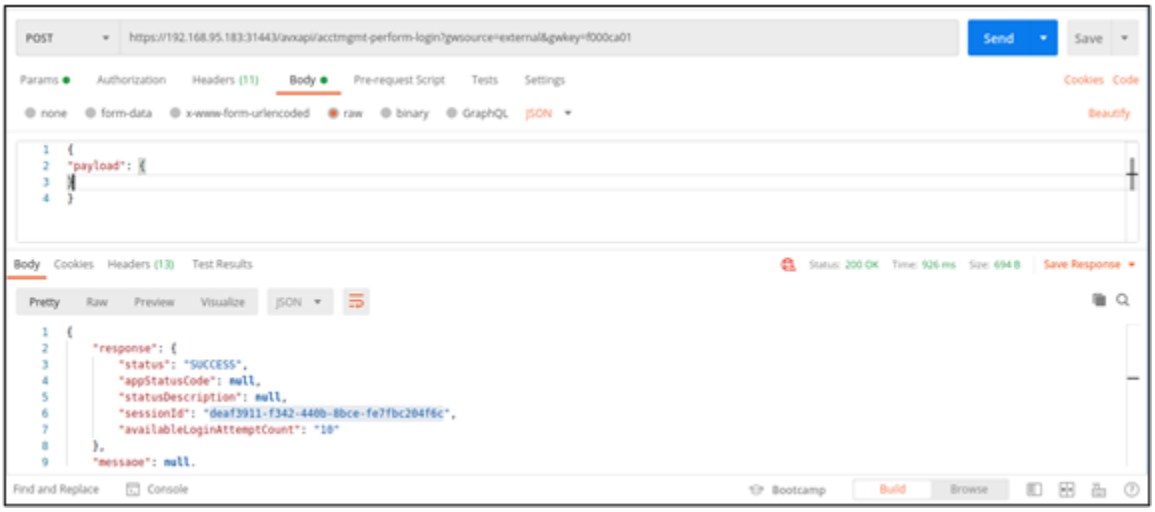
Request Payload

```
{}
```

Sample Response

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
   "response": {
      "status": "SUCCESS",
      "appStatusCode": null,
      "statusDescription": null,
      "sessionId": "avx~~c73a4f56-f4ab-4cdf-aadf-6d90bf406077",
      "authCode": null,
      "lockDownPeriod": 15,
      "emailId": null,
      "termsAccepted": true,
      "passwordExpiryMsg": ""
   },
   "message": "Login successful.",
   "appStatusCode": null,
   "tags": null,
   "headers": null
}
```

What's Next

- **Using Session ID for further API calls**

Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Using Session ID for further API calls

The sessionID retrieved using the login API can be used in the header for making further API calls.

In this section, as an example, we are using the session ID with the API call for adding a role.

Before you begin

- Session ID is obtained from the login API.
- Ensure that the session ID is valid and has not expired.

Request Structure

Endpoint:	`/role`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/role?&gwsource=external`

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

To understand the elements of the sample URL, click here.	
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionId *Header*	(Mandatory) Use session ID retrieved from login API, if username and password are not provided. **Type:** *String* **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external **Type:** *String*
Payload *String*	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
name *String*	(Mandatory) Name of the role to be added. **Example:** "role_1"
description *String*	(Optional) Description of the role to be added. **Example:** "Adding a new role"

Response Structure

- **Status Code:** 201 Created
- **Message:** Role added successfully

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	Contains the response attributes for role added successfully.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Response Message
201 Created	null	Role added successfully.
409 Conflict	ACCT_RO_002	Role name already exists
400 Bad Request	VALIDATION_ERROR_0004	'name' should have at least '2' characters, Mandatory Field 'name' is missing or empty.
400 Bad Request	ACCT_RO_015	Role name invalid.

Sample Request/Response

Use Case

Using the session ID acquired from the login API to execute subsequent API calls, specifically for adding a role API.

Sample Request

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/role?gwsource=external
```

Request Payload

```
{
"payload": {
 "name":"role_01",
 "description":"Adding a new role"
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
}
}
```

Sample Response

```
{
"response": "Role added successfully",
"message": "Role added successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

What's Next

- Search for a Role
- Delete a Role

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Authentication Using a Service Account

For accessing APIs, you can login via two types of accounts:

- User account

  A **User account** represents an individual person interacting with the application or the system. User accounts are used for accessing the system on behalf of a user.

  For accessing APIs with a user account, you need to get the session ID by providing a username and password in the login API. This session ID can then be used for accessing other APIs.

  Note: You can also use the username and password in all API calls instead of the sessionId. However, this is not recommended.

- Service account

  A **Service account** represents a non-human entity such as an application or a service. It is used for automated processes or system-to-system interactions without human intervention.

  For accessing APIs with a service account, you need to get the Access Token by providing Client ID and Client Secret in get-service-token API. This Access Token can then be used for accessing other APIs.

  Note: Access Token Validity is 30 minutes by default and it can be configured in **Settings** > **Authentication** > **oAuth Settings**.

- Retrieve Access Token using get-service-token API
- Using Access Token in the header for further API calls

Retrieve Access Token using get-service-token API

The API provides a streamlined process for retrieving service tokens related to account management tasks.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Before you begin

- Make sure you have valid login credentials for accessing the system.

Request Structure

Endpoint:	`/acctmgmt-get-service-token`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/acctmgmt-get-service-token?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json
Authentication:	Yes
Request timeout period	15 minutes

Input Parameters

	Description
Authorization *Header*	(Mandatory) Please form a string in this format <Client ID>:<Client Secret> and do base64 encoding. Then prepend a key 'Basic' before the encoded value. Final value should be "Basic <EncodedValue>". **Type:** *String* **Example:** "admin"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Type:** *String* **Example:** "application/json"
grant_type *Payload*	(Mandatory) Payload Type should be "Form". The value of the param should be "Client_Credentials".

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

	Description
	Type: *Text*

Response Structure

- **Status Code:** 200 Ok
- **Message:** Successful
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	The response contains the attributes needed to retrieve the access token.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Response Message
200 OK	NA	Successful
400 Bad request	ACCT_SA_003	Service account is invalid/not found::[Service account not found in the database]
400 Bad request	OAUTH_CLNT_015	Client Password is incorrect::[Invalid Client credential]
400 Bad request	ACCT_SA_001	Invalid Request::[Invalid client Id or secret]
500 Internal Server Error	avx-common-011	Error while processing.

Sample Request/Response

Use Case

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Retrieve Access Token.

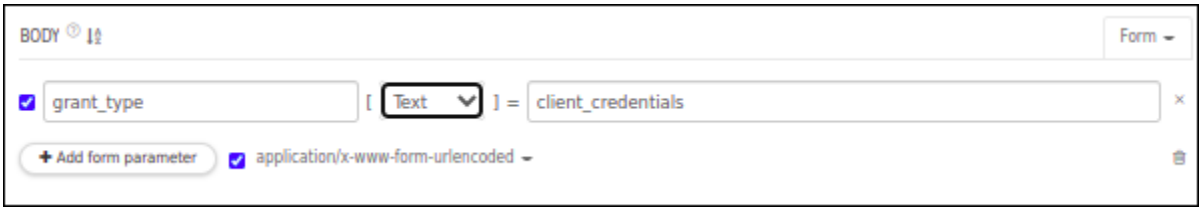
Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/acctmgmt-get-service-token?gwsource=external
```

Content-Type: application/x-www-form-urlencoded

Authorization: Basic
NTIxYzNhZDItZWE0ZS00NDdiLWE1MWItOTYyMWJiN2VhMTI2OjU1QVUjTk84JSpaaGd2TmZhWVVtdHZYMGRrRWhvZ

Request Payload



Sample Response

```
{
"response":
 "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwbGF0Zm9ybSIsImF1ZCI6ImF2eCIsImNsaWVudElkIjoiOTcwNzRlNDEtOGFmOS00NTZkLTlhNjQtZjBj
 NGJiOTA4MDQ4IiwiaXNzIjoiYXZ4IiwiZXhwIjoxNjUwMzY5MzY3LCJncmFudCB0eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIn0.HZnkuUEjXIeqJWqpqiNWFHqIDI7
 GYf4cWx6VwbjGD_0",
"message": null,
"appStatusCode": null,
"tags":{},
"headers": null
}
```

What's Next

- **Using Access Token in the header for further API calls**

Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Using Access Token in the header for further API calls

The access token retrieved using the get-service-token API can be used in the header for making further API calls.

In this section, as an example, we are using the access token with the API call for adding a resource.

Before you begin

- Access Token is obtained from the get-service-token API.
- Ensure that the Access Token is valid and has not expired.

Request Structure

Endpoint:	`/resource`

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/resource?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
Token *Header*	(Mandatory) Use token retrieved from login API. **Type:** *String* **Example:** eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwbGF0Zm9ybSIsImF1ZCI6ImF2eCIsImNsaWVudElkIjoiOTcwNzRlNDEtOGFmOS00NTZkLTlhNjQtZjBjNGJiOTA4MDQ4IiwiaXNzIjoiYXZ4IiwiZXhwIjoxNjUwMzY5MzY3LCJncmFudCB0eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIn0.HZnkuUEjXIeqJWqpqiNWFHqIDI7GYf4cWx 6VwbjGD_0
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external **Type:** *String*
Payload *String*	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
name *String*	(Mandatory) Name of the resource to create. Name cannot be duplicated. **Example:** "resource_1"
description	(Optional) Description of the resource.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
String	**Example:** "This is a sample resource."

Response Structure

- **Status Code:** 201 Created
- **Message:** Resource added successfully
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	Contains the response attributes for resource added successfully.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Response Message
201 Created	null	Resource added successfully
409 Conflict	RBAC_RE_005	Resource with the given name already exists
400 Bad Request	VALIDATION_ERROR_0004	'name' should have at least '2' characters, Mandatory Field 'name' is missing or empty
400 Bad Request	VALIDATION_ERROR_0004	Invalid "name".
401 Unauthorized	AVX_GW_012	Unauthorized access, reason - Invalid Token
407 Proxy Authentication Required	AVX_GW_011	Session validation failed, reason - Session information is missing.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response

Use Case

Add a resource using API with Access Token.

Sample Request

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/resource?gwsource=external
```

Request Payload

```
{
 "payload": {
 "name": "resource_1",
 "description": "This is a sample resource."
 }
 }
```

Sample Response

```
{
 "response": "Resource added successfully",
 "message": "Resource added successfully",
 "appStatusCode": null,
 "tags": null,
 "headers": null
 }
```

What's Next

- Search for a Resource
- Delete a Resource

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **HostName**: A human-readable label assigned to a device (host) on a network

  The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Submit Request

Creates a new workflow request for the given workflow name. A request id is generated and associated with each submitted workflow request which can be used later for retrieving any information about the request.

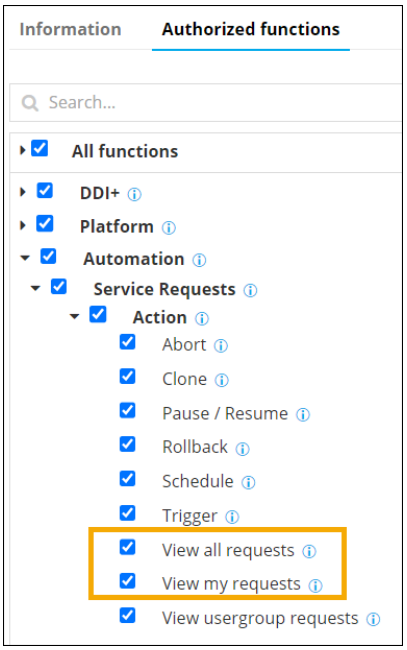
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-submit-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external

Payload
String
	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.
	Any one of the following:
	`Service Task Payload`
	`User Task Payload`

Task Specific Information

Service Task Payload

Service tasks will not have any user input for submitting the request.

Name	Description
workflowName *Header Payload*	(Mandatory) The workflow for which the request is submitted. **Type:** *String* **Example:** "servicenow Test"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response

Use Case

Submit a request for the workflow **servicenow Test**.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external
```

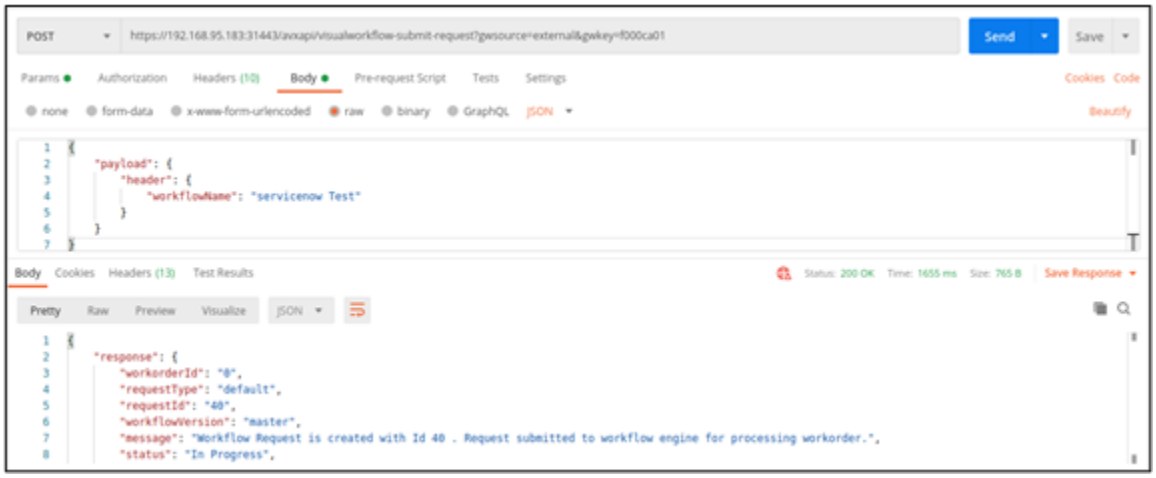
Request Payload

```
{
  "payload": {
    "header": {
      "workflowName": "servicenow Test"
    }
  }
}
```

Sample Response

```
{
  "response": {
    "workorderId": "0",
    "message": "Workflow Request is created with Id 401 . Request submitted to workflow engine for processing workorder.",
    "requestType": "default",
    "requestId": "401",
    "workflowVersion": "master",
    "status": "In Progress",
    "statusCode": 0
  },
  "message": null,
  "appStatusCode": null,
  "tags": null
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



User Task Payload

Name	Description
workflowName	(Mandatory) The workflow for which the request is submitted.
Header Payload	**Type:** *String*
	Example: "Sample 2"
Input Payload	
input	(Mandatory) The input data for the user task.
Data Payload	**Type:** *String*
requestData	(Mandatory) The request data for which the request is submitted.
Request Data Payload	List of **Request Data Payloads**.
Request Data Payload	
scenario	(Mandatory) The scenario name for the form.
Request Data Payload	**Type:** *String*
	Example: "scenario"
sequenceNo	(Mandatory) The sequence number of the scenario.
Request Data Payload	**Type:** *Integer*

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
	Example: "2"
fieldInfo *Request Data Payload*	(Mandatory) The field info configured in the form. The value should be key value, field id and its value. **Type:** *Json* **Example:** "deviceName": "ADC"
task_action *Data Payload*	(Mandatory) Action to be performed on the task. **Type:** *Integer* **Constraints:** 1 for success case and 2 for reject case. **Example:** "1"

Sample Request/Response - Form Task

Use Case

Submit a request for the workflow with the first task as **Form**.

Note: When using the visualworkflow-submit-request API for a form task, any regex pattern provided by the user will be validated on the server side. It is essential that users provide a valid value in the payload.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external
```

Request Payload

```
{
  "payload": {
    "header": {
      "workflowName": "Sample 2"
    },
    "data": {
      "input": {
        "requestData": [
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
        {
          "scenario": "scenario",
          "sequenceNo": 1,
          "fieldInfo": {
            "deviceName": "ADC"
          }
        },
        {
          "scenario": "scenario2",
          "sequenceNo": 2,
          "fieldInfo": {
            "deviceName": "A10"
          }
        }
      ]
    },
    "task_action": 1
  }
 }
}
```

Sample Response

```
{
  "response": {
    "workorderId": "0",
    "message": "Workflow Request is created with Id 55 . Request submitted to workflow engine for processing workorder.",
    "requestType": "default",
    "requestId": "401",
     "workflowVersion": "master",
    "status": "In Progress",
    "statusCode": 0
  },
  "message": null,
  "appStatusCode": null,
  "tags": null
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response - Grid Task

Use Case

Submit a request for the workflow with the first task as **Grid**.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external
```

Request Payload

```
{
"payload": {
   "data": {
    "input": {
      "columnLabels": {
        "IP": "IP",
        "Device Name": "Device Name"
      },
      "gridData": [
        {
          "IP": "","Device Name": ""
        }
      ]
    },
    "globalData" : { "gridData": [
      {
```

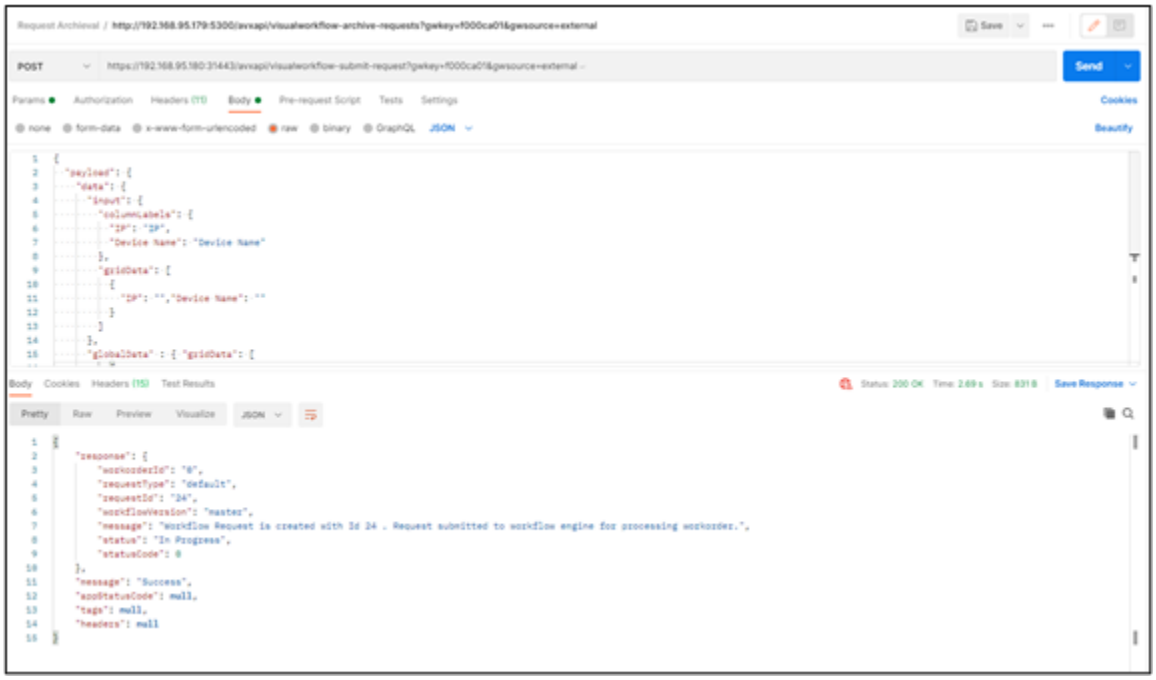
```
          "IP": "192.168.15.35",
          "Device Name": "Device F5"
        },
        {
          "IP": "192.168.15.40",
          "Device Name": "Device AVI"
        }
      ]},
    "task_action": 1
  },
  "header": {
    "workflowName": "Grid"
  }
 }
}
```


Note: Global Variable `<%gridData%>` should be present in the “values” section in Grid Task.

Sample Response


```
{
 "response": {
   "workorderId": "0",
   "message": "Workflow Request is created with Id 24 . Request submitted to workflow engine for processing workorder.",
   "requestType": "default",
   "requestId": "401",
    "workflowVersion": "master",
   "status": "In Progress",
   "statusCode": 0
 },
 "message": null,
 "appStatusCode": null,
 "tags": null
}
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Sample Request/Response - Script Task

Use Case

Submit a request for the workflow with the first task as **Script**.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external
```

Request Payload

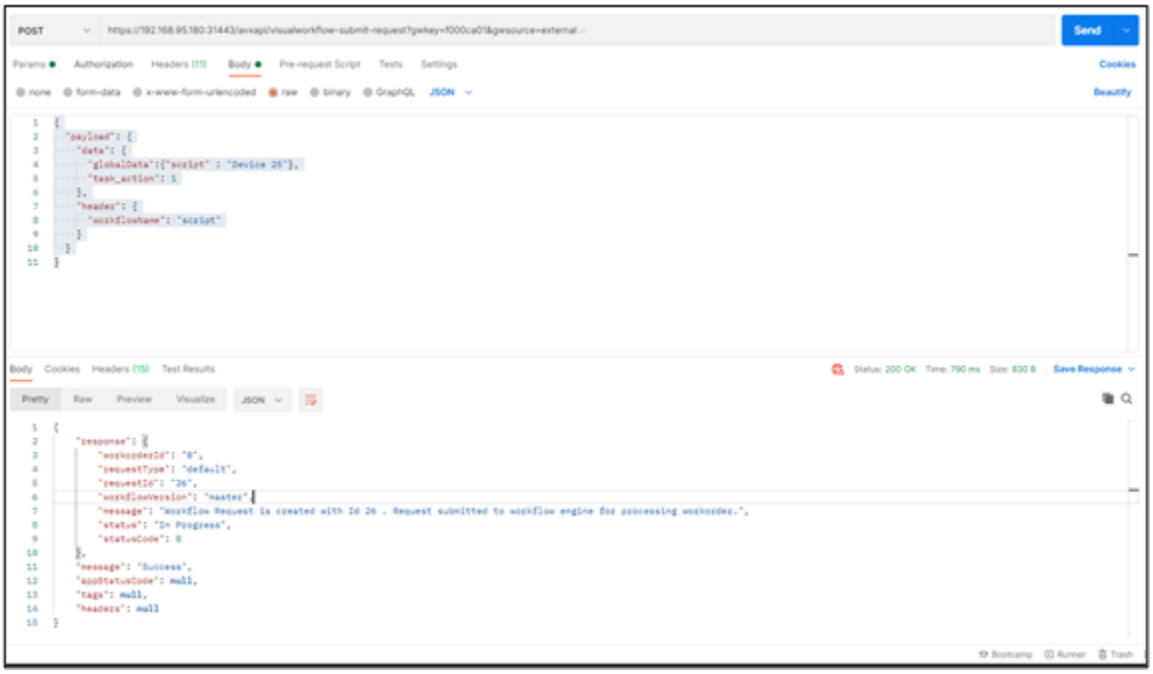
```
{
  "payload": {
    "data": {
      "globalData":{"script" : "Device 25"},
      "task_action": 1
    },
    "header": {
      "workflowName": "script"
    }
  }
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: Values can be passed to available Global variables present inside the Script task.

Sample Response


```
{
 "response": {
  "workorderId": "0",
  "message": "Workflow Request is created with Id 26. Request submitted to workflow engine for processing workorder.",
  "requestType": "default",
  "requestId": "401",
   "workflowVersion": "master",
  "status": "In Progress",
  "statusCode": 0
 },
 "message": null,
 "appStatusCode": null,
 "tags": null
}
```

Sample Request/Response - YAML Task

Use Case

Submit a request for the workflow with the first task as **YAML**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-submit-request?gwsource=external
```

Request Payload

```
{
 "payload": {
  "data": {
   "input": {
    "editorContent": "<%yaml%>",
    "type": "YAML"
   },
   "globalData":{"yaml" : "---\n-\n    name: Manage BIG-IP APM policy or APM access profile imports\n    hosts: localhost\n    tasks:\n        - name: Import APM
profile\n          bigip_apm_policy_import:\n            name: new_apm_profile"},
   "task_action": 1
  },
  "header": {
   "workflowName": "yaml"
  }
 }
}
```

Note: Global Variable `<%yaml%>` should be present in the YAML task.

Sample Response

```
{
 "response": {
  "workorderId": "0",
  "message": "Workflow Request is created with Id 26 . Request submitted to workflow engine for processing workorder.",
  "requestType": "default",
  "requestId": "401",
   "workflowVersion": "master",
  "status": "In Progress",
  "statusCode": 0
 },
 "message": null,
 "appStatusCode": null,
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
  "tags": null
}
```

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response params for the search object request. For more response details.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response, will be non-null for failure response.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
tags *NA*	More info in case of failure response.

Response

Name	Description
requestID *String*	Unique identifier for the request.
workflowVersion *String*	Version of the workflow.
message *String*	The message with the status and request number for the request.
status *String*	Status of the request.
statusCode *Integer*	Status code of the request
requestType *String*	The type of the request.
workorderId *String*	The workoderId for the request. If there is no workorder the value will be 0.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
400 Bad request	avx-common-028	Invalid/Incorrect payload. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input.
401 Unauthorized	WORKFLOW_1679	User is not authorized. **Possible Remediation:** LoggedIn user is not authorized to access the workflow.
404 Not Found	engine-db-015	Workflow not found. **Possible Remediation:** Invalid workflow data.
409 Conflict	WORKFLOW_1475	Given workflow is not in enabled state. **Possible Remediation:** Given workflow is not in enabled state.
500 Internal Server Error	WORKFLOW_1617	Invalid form data. Please provide all mandatory data. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input.
500 Internal Server Error	NA	Error while processing. **Possible Remediation:** • Task is already completed. • Task is being processed by another user.

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **HostName**: A human-readable label assigned to a device (host) on a network

  The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Update Request

Updates an existing request waiting for a user input to progress and proceed to the next task of the workflow. This is applicable only for tasks that need manual intervention. For example, Form task.

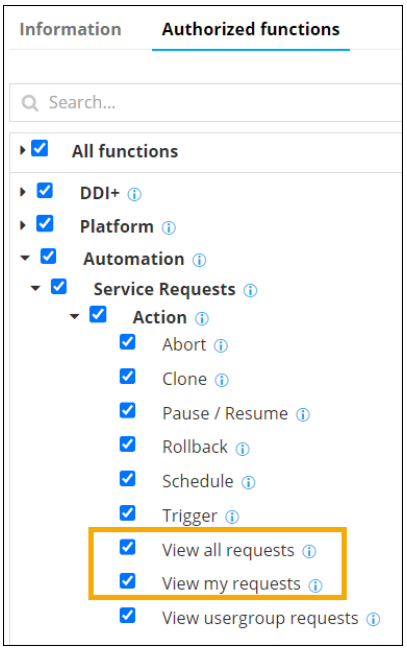
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-update-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *String*	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
Header Payload	
request_id *String*	(Mandatory) The request ID to update the request details. **Example:** "409"
workorder_id *String*	(Mandatory) The workorder ID to update the request details. **Constraints:** If there is no workorder in the workflow then the default value will be zero (0). **Example:** "1"
task_id	(Mandatory) The task ID which will be updated.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
String	**Example:** "reviewComponent_1"
Data Payload	
input *Data Payload*	(Mandatory) The input data for the user task. **Type:** *String*
task_action *Data Payload*	(Mandatory) Action to be performed on the task. **Type:** *Integer* **Constraints:** 1 for success case and 2 for reject case. **Example:** "1"
Input Payload	
config *Review Task Payload*	(Mandatory) The prevalidation, implementation, post validation, and rollback configuration for the devices. **Type:** *Object*
gridData *Grid Task Payload*	(Mandatory) The grid configuration to be updated in the task. **Type:** *Object*

Task Specific Information

Sample Request/Response - Review Task

Use Case

Update a request with a **Review** task.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-request?gwsource=external
```

Request Payload

```
{
  "payload": {
    "header": {
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"request_id": "936",
      "workorder_id": "0",
      "task_id" : "reviewComponent_1"
 },
 "data":{
  "input" : {"config": {
 "implement": {
   "deviceList": [
     {
       "deviceName": "192.168.41.101",
       "commands": [
         {
           "command": "<push></push>",
           "deviceName": "",
           "uniqueNumber": 2,
           "sequenceNumber": 0,
           "properties": {
             "sleep": 0
           },
           "type": "push"
         }
       ]
     }
   ]
 },
 "prevalidation": {
   "deviceList": [
     {
       "deviceName": "192.168.41.101",
       "commands": [
         {
           "command": "<push></push>",
           "deviceName": "",
           "uniqueNumber": 2,
           "sequenceNumber": 0,
           "properties": {
             "sleep": 0
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
            },
            "type": "push"
          }
        ]
      }
    ]
  },
  "postvalidation": {
    "deviceList": [
      {
        "deviceName": "192.168.41.101",
        "commands": [
          {
            "command": "<push></push>",
            "deviceName": "",
            "uniqueNumber": 2,
            "sequenceNumber": 0,
            "properties": {
              "sleep": 0
            },
            "type": "push"
          }
        ]
      }
    ]
  }
 }
 },


    "task_action":1
  }
 }
}
```

Sample Response

```
{
 "response": {
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
  "message": "Request submitted to workflow engine for processing workflow request 936",
  "requestId": "936"
 },
 "message": null,
 "appStatusCode": null,
 "tags": null
}
```


Sample Request/Response - Review Task within a Subflow

Use Case

Update a request with a review palette, where the **Review** task is within a subflow.

Request URL


```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-request?gwsource=external
```


Request Payload


```
{
 "payload": {
  "header": {
 "request_id": "936",
      "workorder_id": "0",
      "task_id" : "reviewComponent_1#subflowid"
  },
  "data":{
   "input" : {"config": {
  "implement": {
   "deviceList": [
    {
     "deviceName": "192.168.41.101",
     "commands": [
      {
       "command": "<push></push>",
       "deviceName": "",
       "uniqueNumber": 2,
       "sequenceNumber": 0,
       "properties": {
        "sleep": 0
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
          },
          "type": "push"
        }
      ]
    }
  ]
},
"prevalidation": {
  "deviceList": [
    {
      "deviceName": "192.168.41.101",
      "commands": [
        {
          "command": "<push></push>",
          "deviceName": "",
          "uniqueNumber": 2,
          "sequenceNumber": 0,
          "properties": {
            "sleep": 0
          },
          "type": "push"
        }
      ]
    }
  ]
},
"postvalidation": {
  "deviceList": [
    {
      "deviceName": "192.168.41.101",
      "commands": [
        {
          "command": "<push></push>",
          "deviceName": "",
          "uniqueNumber": 2,
          "sequenceNumber": 0,
          "properties": {
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
            "sleep": 0
          },
          "type": "push"
        }
      ]
    }
   ]
  }
 }
 },


   "task_action":1
  }
 }
}
```

Sample Response

```
{
 "response": {
  "message": "Request submitted to workflow engine for processing workflow request 936",
  "requestId": "936"
 },
 "message": null,
 "appStatusCode": null,
 "tags": null
}
```

Grid Task Payload

Service tasks will not have any user input for submitting the request.

Sample Request/Response - Grid Task

Use Case

Update a request with a **Grid** task.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-request?gwsource=external
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

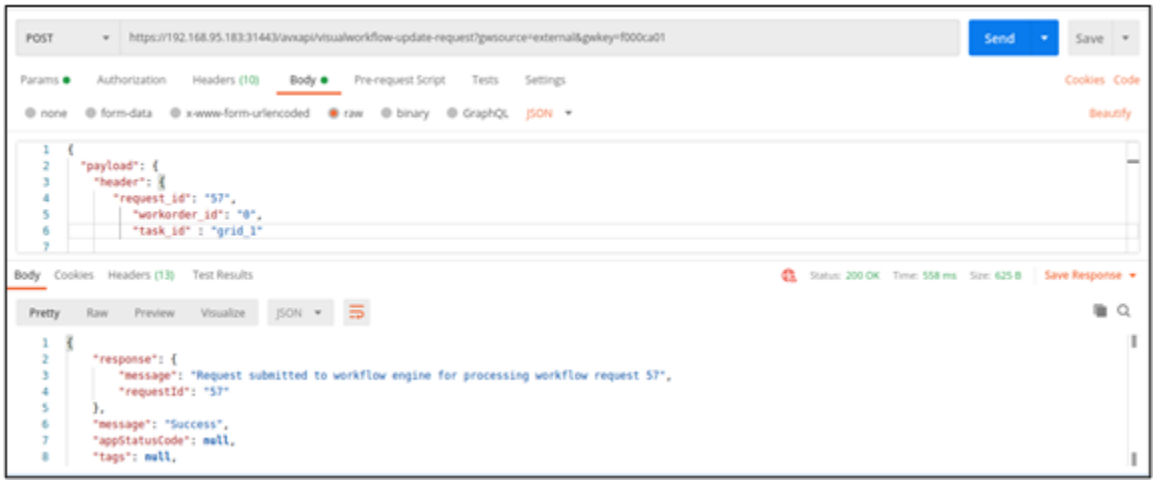
Request Payload


```
{
  "payload": {
 "header": {
   "request_id": "937",
   "workorder_id": "0",
   "task_id": "gridComponent_1"
 },
 "data": {
  "input": {
    "gridData": [
      {
        "name": "Appviewx",
        "designation": "Network"
      },
      {
        "name": "ADC",
        "designation": "Device"
      }
    ]
  },
  "task_action": 1
 }
 }
```


Sample Response


```
{
  "response": {
    "message": "Request submitted to workflow engine for processing workflow request 936",
    "requestId": "936"
  },
  "message": null,
  "appStatusCode": null,
  "tags": null
}
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response params for the search object request.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response, will be non-null for failure response.
tags *NA*	More info in case of failure response.
requestId *String*	Unique identifier for the request.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
message *String*	The message with the status and request number for the request.

Status Codes

HTTP Status Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad request	avx-common-028	Invalid/Incorrect payload. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Invalid workflow data.
500 Internal Server Error	WORKFLOW_1056	Workflow request already closed. **Possible Remediation:** Ensure workflow is not closed.
500 Internal Server Error	NA	Internal Server Error. **Possible Remediation:** • Task is already completed. • Task is being processed by another user. • Service Request is paused. • Service Request has been aborted. • Service request is closed. • Given task is not the right task to be executed with Id.

Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get Current Task Detail of the Request

To get the details of the current active task of a specific workflow request. An action can be performed only on the current active task of a workflow.

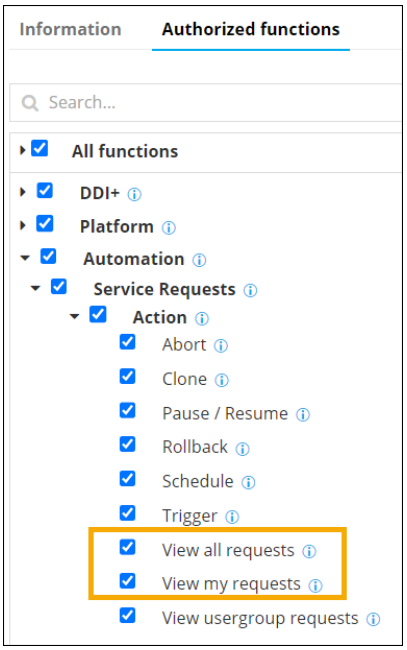
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-get-request-current-task-details
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-request-current-task-details?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *String*	(Mandatory) Contains all the params to be sent in the request body for the post request. Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
request_id *String*	(Mandatory) The request ID to update the request details. **Example:** "409"
workorder_id *String*	(Mandatory) The workorder ID to update the request details. If there is no workorder in the workflow then the default value will be zero (0). **Example:** "1"
task_id *String*	(Mandatory) The task ID which will be updated. **Example:** "reviewComponent_1"
User Task Payload	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
Header Payload	(Mandatory) Contains header param details of the request
Data Payload	(Mandatory) Contains params for the user task

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	avx-common-028	Invalid/ Incorrect payload. **Possible Remediation:** Check and ensure if a valid value is given in request payload field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Check and ensure if a valid requestId/workorderId is given.
500 Internal Server Error	engine-db-022	Unable to fetch task status. **Possible Remediation:** Check and ensure if a valid task ID is given.

Sample Request/Response

Use Case

To get the current task details for a request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-request-current-task-details?gwsource=external
```

Request Payload

```
{
  "payload": {
    "requestId": "409",
    "workorderId": "1",
    "taskId": "reviewComponent_1"
  }
}
```

Sample Response

```
{
  "response": {
        "jobId": null,
        "requestStatus": 0,
        "createdTime": 0,
        "request_id": "409",
        "workorder_id": "1",
        "workorder_status": "Completed",
        "workflowName": "Sample",
    "data": {
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

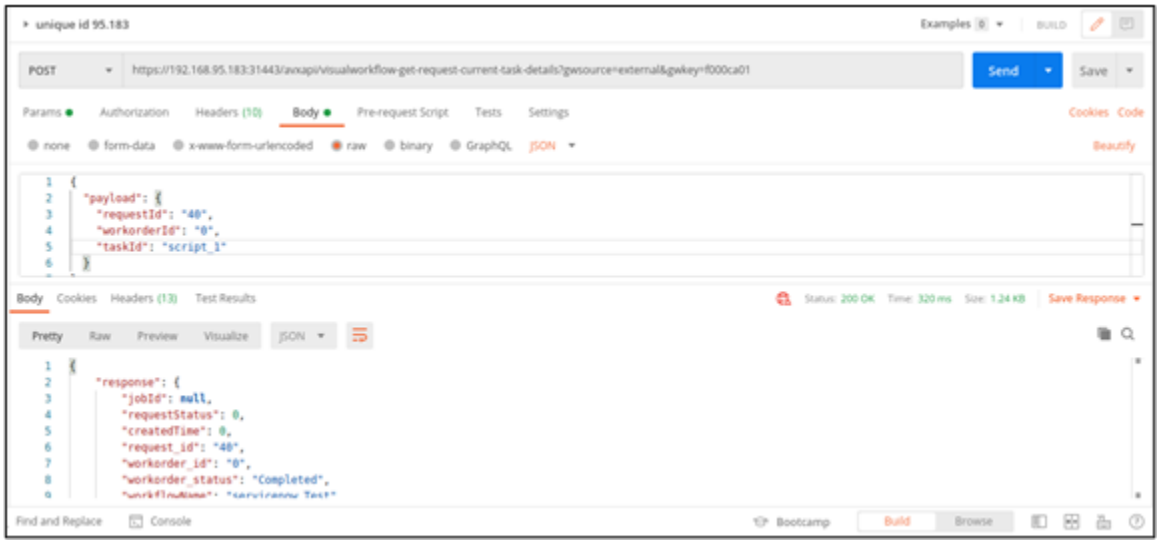
```
"comments": "Hi Tester of RC",
"state": 1,
"taskStatus": "Success",
"task_id": "reviewComponent_1",
"task_name": "Review Component",
"input": {

},
"output": {
  "comments": "Hi Tester of RC",
  "configLabel": {
    "Prevalidation": "Pre Validation",
    "implementation": "Implementation",
    "PostValidation": "Post Validation"
  },
  "enablePrevalidation": true,
  "fetchOnLoad": true,
  "proceedProcess": "Submit",
  "ticketDetails": {
    "ticketNumber": "",
    "startTime": null,
    "endTime": null
  },
  "configureChangeManagement": true,
  "isAuthorized": true,
  "vendor": "CHANGE",
  "breakdownProcess": "Cancel",
  "readWrite": true,
  "prevalidationValue": "Check",
  "config": {
    "Prevalidation": {
      "deviceList": [
        {
          "deviceName": "f5",
          "commands": [
            "tmsh",
            "list gtm pool pool41_161_demo"
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
          ]
        }
      ]
    },
    "implementation": {
      "deviceList": [
        {
          "deviceName": "f5",
          "commands": [
            "tmsh",
            "create gtm pool pool41_161_demo"
          ]
        }
      ]
    },
    "PostValidation": {
      "deviceList": [
        {
          "deviceName": "f5",
          "commands": [
            "tmsh",
            "show sys version"
          ]
        }
      ]
    }
  },
  "logs": [
    "Request doesn't have valid ticket details."
  ],
  "implementationType": "Manual"
},
"is_picked": false,
"roles": null,
"tags": null,
"label": null,
"component_type": "review",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "task_process_state": 1,
    "is_completed": true,
    "task_category": "review"
  }
 },
 "message": null,
 "appStatusCode": null,
 "tags": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get Request Summary/Log

Get the request summary/logs details of a particular workflow request. The execution details of the workflow can be viewed from these logs.

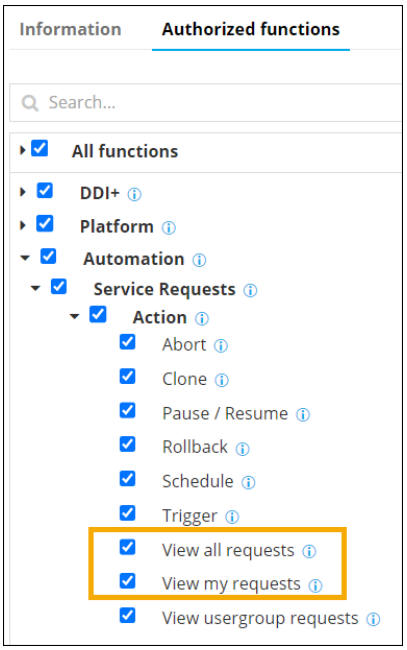
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-request-logs
Type:	GET
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-request-logs?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
ids *Query*	(Mandatory) Request IDs which is a string value. **Type:** *String*

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message	Success message or failure description in case of error.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
String	
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	WORKFLOW_1300	Workflow Request ID is Mandatory. **Possible Remediation:** Check and ensure if a valid request ID is given.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Invalid workflow data

Sample Request/Response

Use Case

To get the request summary/logs for a request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-request-logs?gwsource=external
```

Request Payload

```
NA
```

Sample Response

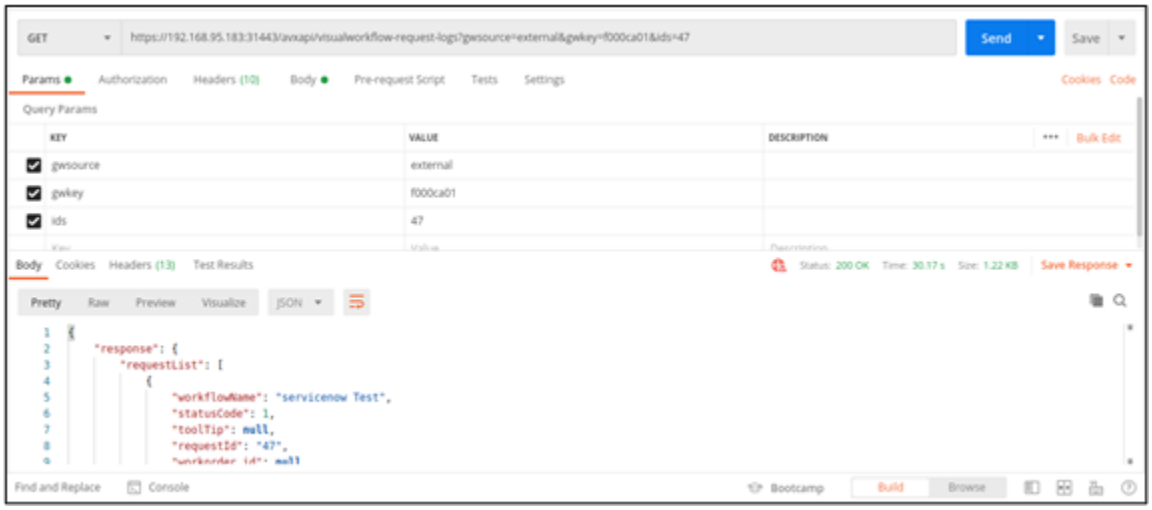
```
{
   "response": {
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"workOrderList": [
  {
    "statusCode": 1,
    "requestId": "410",
     "created_by": "admin",
    "status": "Completed",
    "workflowName": "Copy of reviewComponent Test",
     "toolTip": null,
    "workorder_id": null,
     "description": null,
    "ticket_details": null,
    "tasks": [
      {
        "state": 1,
       “comments”:null,
        "task_name": "WorkOrder",
        "task_id": "startWorkOrder_1",
        "user": "admin",
        "task_status": "Success",
        "task_start_time": 1502696313667,
        "task_end_time": 1502696729781,

        "logs": [
          {
            "time": 1502696313815,
            "message": "Child WorkOrder ID: 1, has been created for the Child workflow: WorkOrder"
          },
          {
            "time": 1502696729781,
            "message": "WorkOrder Completed"
          }
        ]
      }
    ],
    "created_time": 1502696305239,
    "rollback_reference_id": "",
    "clone_reference_ids": [],
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
            "last_updated": 1502696305239,
            "runnning_time": 0
        }
    ]
  },
  "message": null,
  "appStatusCode": null,
  "tags": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get Associated Workorder

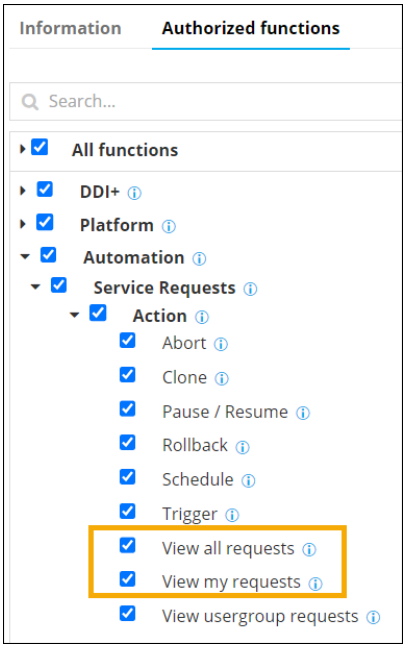
Retrieve the workorders associated with a specific workflow request. Workorders are used when there are multiple sub workflows involved with a workflow. Each workorder takes a separate parallel execution path.

Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Structure

Endpoint:	`/visualworkflow-get-associated-workorder`
Type:	`GET`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-associated-workorder?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
requestId *Query*	(Mandatory) Request ID which is a string value. **Type:** *String*

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	NA	Workflow Request ID is Mandatory.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
		Possible Remediation: Check and ensure if a valid request ID is given.
404 Not Found	NA	Request not found. **Possible Remediation:** Check and ensure if a valid request ID is given.

Sample Request/Response

Use Case

To retrieve the workorders associated with a specific workflow request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-associated-workorder?gwsource=external
```

Request Payload

```
NA
```

Sample Response


```
{
  "response": {
    "workorderList": [
      {
    "workorderId": "1",
    "createdTime": 1528957336385,
    "lastUpdatedTime": 1528957336385,
    "description": "startWorkOrder_1",
    "toolTip": {


    },
    "log": [
      {
        "time": 1528957341359,
        "message": "Initiating Script"
      }]
  }],
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
 "iTotalDisplayRecords": 1
 }
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get Tasks which are in InProgress State

Retrieve all the tasks that are in progress for a specific request. When there are sub workflows involved there could be more than one active in-progress tasks.

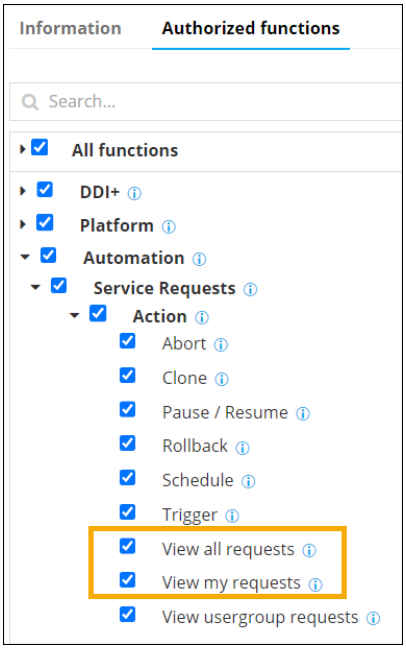
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Request Structure

Endpoint:	/visualworkflow-get-tasks-inprogress
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-tasks-inprogress?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
request_id *String*	(Mandatory) The request ID to update the request details. **Example:** "2"
workorder_id	(Mandatory) The workorder ID to update the request details.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
String	**Constraints:** If there is no workorder in the workflow then the default value will be zero (0). **Example:** "0"

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	avx-common-028	Invalid/Incorrect payload.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
		Possible Remediation: Check and ensure if a valid value is given in request payload field.
400 Bad request	common-service-007	Input data is not valid. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input requestId/workorderId.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input requestId/workorderId.

Sample Request/Response

Use Case

To retrieve all the tasks of a request, which are in Inprogress state.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-tasks-inprogress?gwsource=external
```

Request Payload

```
{
 "payload": {
 "requestId": "2",
 "workorderId": 0
}
}
```

Sample Response

```
{
 "response": {
"tasksInProgress": [
  {
    "status": "In Progress",
    "taskName": "Implementation"
  }
]
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
 }
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Upload File

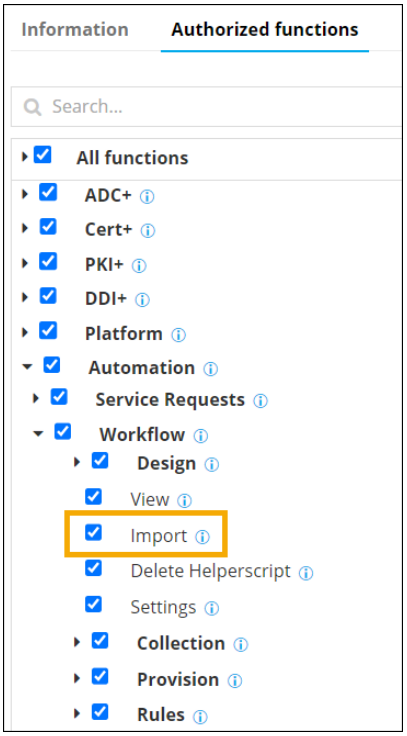
Upload file supports uploading a file and overwriting an uploaded file.

Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Automation** > **Workflow** > **Import**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Request Structure

Endpoint:	/visualworkflow-file-operation-upload
Type:	POST

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-upload?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Payload

Name	Description
fileName *String*	(Mandatory) Name of the file, allowed file formats : img, jpg, jpeg, png, doc, pdf, txt, gif, zip, ppt, pptx, xls, xlsx, csv, p7b, p7c, pfx, p12, der, pem, csr, crt, cer, key, xml. **Example:** "sample.zip"
uniqueId *String*	(Mandatory) Unique ID for the file being uploaded. **Example:** "651baff382ca812a7cbf4baa"
fileOverWrite *Boolean*	(Mandatory) Determines whether to overwrite the file being uploaded if a file with the same uniqueId exists. False (Upload file only if a file with same uniqueId not present.) True (Upload file and overwrite if file with same uniqueId is present. If not, then upload the given file.) **Example:** "false/true"
file *Binary*	(Mandatory) File to be uploaded. **Example:** "<%File to be uploaded%>"
enableDownloadLimit *Boolean*	Determines whether to enable the download limit. The default value is False False (Download limit will be disabled. File can be download the file any number of times) True (Enable the download limit)
maxDownloadLimit *Integer*	The maximum number of times user is allowed to download the uploaded file. If **enableDownloadlimit** is true, and no value is provided in the payload then the default value will be 1. The value is limited to a mimimum of 1 and maximum of 10.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure

- **Status Code: 200 OK**
- **Message: File Uploaded successfully**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
500 Internal Server Error	WORKFLOW_1000	Invalid input from the client. **Possible Remediation:** The payload is invalid, check and update it.
500 Internal Server Error	StudioService_080	Error while uploading file.
400 Bad Request	StudioService_078	UniqueId field not found.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
		Possible Remediation: Check and ensure if valid uniqueId is given.
400 Bad Request	StudioService_115	Invalid maxDownloadLimit. **Possible Remediation:** Ensure the maxDownloadLimit value is between the range 1 to 10.
400 Bad Request	StudioService_077	File content not found. **Possible Remediation:** Check and ensure for valid file.
400 Bad Request	StudioService_071	Invalid file name. **Possible Remediation:** Check and ensure if valid fileName is given.
400 Bad Request	StudioService_076	File Name not found. **Possible Remediation:** Check and ensure if valid fileName is given
400 Bad Request	StudioService_040	Invalid field format **Possible Remediation:** Check and ensure if valid file and fileName is given.
400 Bad Request	StudioService_075	File with uniqueId already exists.

Sample Request/Response

Use Case 1: Upload file

Upload file to collection.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-upload?gwsource=external
```

Request Payload

```
{
  "payload": {
 “fileName”:”sample.zip”,
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
 “uniqueId”:”uniqueId”,
 “fileOverWrite”:false,
 “file”:<%File to be uploaded%>
  }
}
```

Sample Response

```
{
  "response": "File Uploaded successfully",
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Use Case 2: Overwrite file

Overwrite file: Upload a file and overwrite if a file with the same uniqueID is present in the database.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-upload?gwsource=external
```

Request Payload

```
{
  "payload": {
 “fileName”:”sample.zip”,
 “uniqueId”:”uniqueId”,
 “fileOverWrite”:true,
 “file”:<%File to be uploaded%>
  }
}
```

Sample Response

```
{
  "response": "File Uploaded successfully",
  "message": "File Uploaded successfully",
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Download File

API to download the uploaded file from the database.

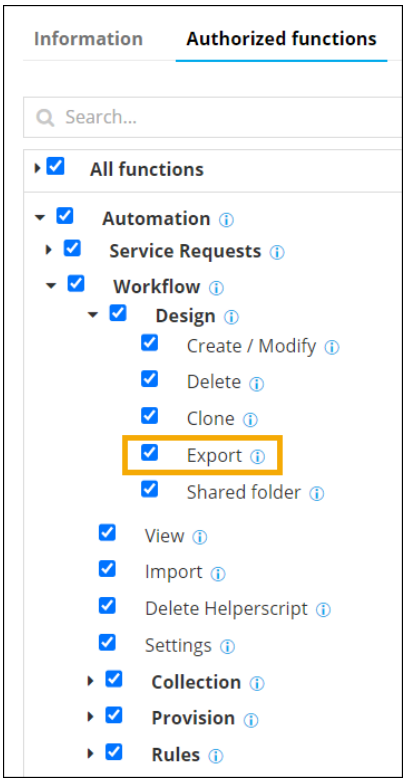
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Automation** > **Workflow** > **Export**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-file-operation-download
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-download?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
uniqueId *String*	(Mandatory) UniqueId of the file to be deleted. **Example:** "uniqueId"
decodeFile *Boolean*	(Optional) Decide whether the file is to be decoded while downloading.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure

- **Status Code: 200 OK**
- **Message: File Downloaded Successful**
- **Headers:**
  - **Content-Type:** application/json
- **Response:** File downloaded successfully.

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
500 Internal Server Error	WORKFLOW_1000	Invalid input from the client. **Possible Remediation:** The payload is invalid, check and update it.
500 Internal Server Error	StudioService_079	Error occured while downloading file. **Possible Remediation:** Exception might occurs while downloading file from DB.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
404 Not Found	StudioService_082	File not found in db. **Possible Remediation:** File with given uniqueId may not exists in DB.
400 Bad request	StudioService_078	UniqueId field not found. **Possible Remediation:** Check and ensure if valid uniqueId is given in the payload.
400 Bad Request	StudioService_111	File download limit exceeded.

Sample Request/Response

Use Case

To download the uploaded file from the database.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-download?gwsource=external
```

Request Payload

```
{
 "payload": {
“uniqueId”:“uniqueId” }
}
```

Sample Response

```
File will be downloaded
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Delete File

API to delete an uploaded file from the database.

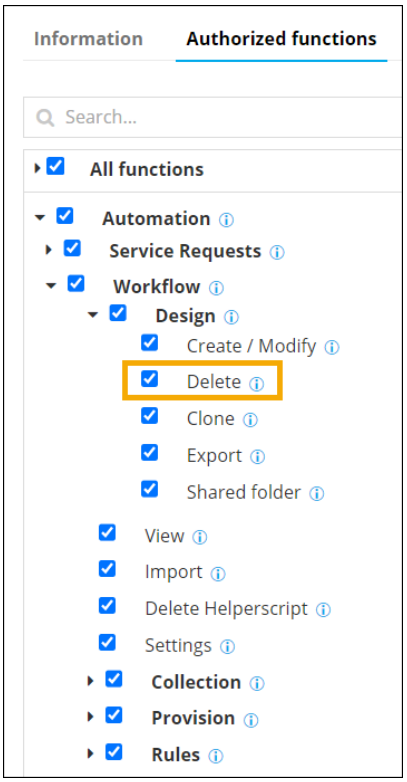
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- Go to **Automation** > **Workflow** > **Design** > **Export**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-file-operation-delete
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-delete?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
Password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
uniqueId *String*	(Mandatory) UniqueId of the file to be deleted. **Example:** "uniqueId"

Response Structure

- **Status Code: 200 OK**
- **Message: File Deleted successfully**
- **Headers:**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Content-Type:** application/json

• **Response:** File Deleted successfully.

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success : File Deleted successfully.
500 Internal Server Error	WORKFLOW_1000	Invalid input from the client **Possible Remediation:** The payload is invalid, check and update it.
500 Internal Server Error	StudioService_081	Error occurred while deleting file. **Possible Remediation:** Chances of file not available in DB or some mongoDb issue. Kindly check logs.
400 Bad request	StudioService_078	UniqueId field not found. **Possible Remediation:** Check and ensure if valid uniqueId is given.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response

Use Case

To delete an uploaded file from the database.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-file-operation-delete?gwsource=external
```

Request Payload

```
{
  "payload": {
 “uniqueId”:“uniqueId” }
}
```

Sample Response

```
{
  "response": "File Deleted successfully",
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Abort Request

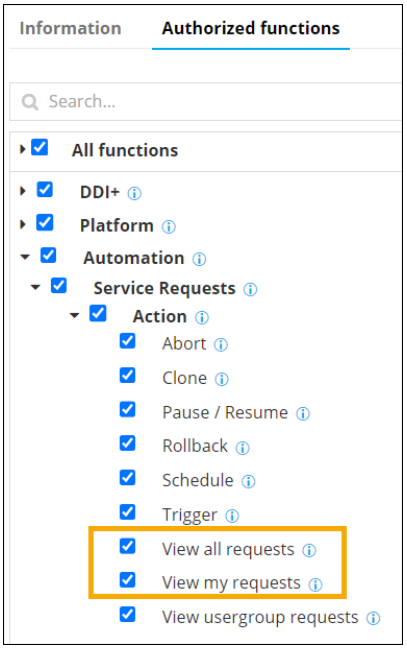
Abort a particular workflow request.

Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**
- **Automation** > **Service Requests** > **Action** > **Abort**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Structure

Endpoint:	`/visualworkflow-abort-request`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-abort-request?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API; it is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username; it is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password; it is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) Specifies the nature of the data in the payload. The parameter should be set to `application/json`. **Example:** "application/json"
gwkey *Query*	Tenant key; needed only in case of multi tenant installations.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
requestId *Payload* *Query*	(Mandatory) Enter the request ID. **Type:** *String* **Example:** "ce7f1a14-2bf9-4e4a"

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	avx-common-028	Invalid/ Incorrect payload. **Possible Remediation:** Check and ensure if a valid request id is given.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Check and ensure if a valid request id is given.

Sample Request/Response

Use Case:

To abort a particular request using request id.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-abort-request?gwsource=external
```

Request Payload

```
NA
```

Sample Response

```
{
  "response": null,
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Clone Request

Clone a particular workflow request.

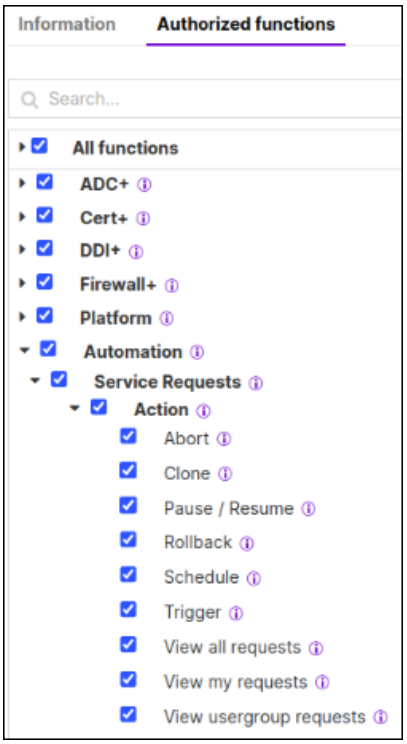
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**
- **Automation** > **Service Requests** > **Action** > **Clone**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-clone-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-clone-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
requestId *Payload* *map*	(Mandatory) Enter the request ID. **Type:** *String* **Example:** "19"

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	Contains the response object along with the hierarchy of the object mapped in field children.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
Key value pair	
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	avx-common-028	Invalid/ Incorrect payload. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Check and ensure if a valid requestId/workorderId is given.

Sample Request/Response

Use Case:

To clone a particular request using request id.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-clone-request?gwsource=external
```

Request Payload

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
   "header": {
      "cloneRequestId": "19"
   }
}
```

Sample Response

```
{
   "response": {
      "workorderId": "0",
      "requestType": "default",
      "requestId": "21",
      "workflowVersion": "master",
      "message": "Workflow Request is created with Id 21 . Request submitted to workflow engine for processing workorder.",
      "status": "In Progress",
      "statusCode": 0
   },
   "message": null,
   "appStatusCode": null,
   "tags": null,
   "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get All Requests

Get all the available workflow requests.

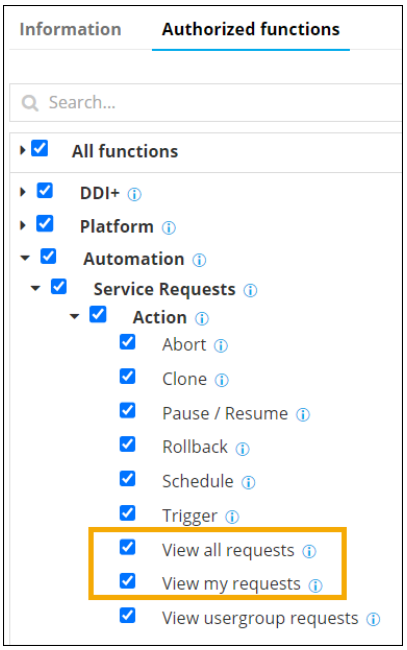
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-get-all-requests
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-all-requests?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external

Response Structure

- **Status Code: 200 OK**
- **Message:** Success
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
String	
tags	More info in case of failure response.
NA	

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	common-service-011	Invalid maxSize **Possible Remediation:** Check whether the maximum size is less than or equal to 100.

Sample Request/Response

Use Case:

To get all the available workflow requests.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-get-all-requests?gwsource=external
```

Request Payload

```
{
   "payload": {
      "startIndex": 0,
      "maxSize": 25,
      "sortColumn": "",
      "sortOrder": ""
   }
}
```

Sample Response

```
{
   "response": {
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"requestList": [
    {
        "workflowName": "Sample Workflow",
        "statusCode": 0,
        "toolTip": {},
        "requestId": "19",
        "workorder_id": "0",
        "created_by": "admin",
        "status": "In Progress",
        "created_time": 1713955000978,
        "last_updated": 1713955001242
    },
    {
        "workflowName": "Citrix NetScaler Golden Config Compliance",
        "statusCode": 1,
        "toolTip": {},
        "requestId": "18",
        "workorder_id": "0",
        "created_by": "system",
        "status": "Completed",
        "created_time": 1713915064190,
        "last_updated": 1713915070287
    },
    {
        "workflowName": "F5 BIG-IP Golden Config Compliance",
        "statusCode": 1,
        "toolTip": {},
        "requestId": "17",
        "workorder_id": "0",
        "created_by": "system",
        "status": "Completed",
        "created_time": 1713915003097,
        "last_updated": 1713915010309
    },
    {
        "workflowName": "F5 BIG-IP CVE Reporting",
        "statusCode": 1,
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "toolTip": {},
    "requestId": "16",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713915002872,
    "last_updated": 1713915009482
},
{
    "workflowName": "Notify Unused VIP-State|Status|Statistics",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "15",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713915002840,
    "last_updated": 1713915011631
},
{
    "workflowName": "F5 BIG-IP CVE Reporting Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "14",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713915001293,
    "last_updated": 1713915004428
},
{
    "workflowName": "Unused Objects Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "12",
    "workorder_id": "0",
    "created_by": "system",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "status": "Completed",
    "created_time": 1713915001280,
    "last_updated": 1713915004368
},
{
    "workflowName": "F5 and Citrix Golden Config Compliance Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "13",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713915001278,
    "last_updated": 1713915065187
},
{
    "workflowName": "Fetch F5 BIG-IP CVEs",
    "statusCode": 2,
    "toolTip": {},
    "requestId": "11",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Failed",
    "created_time": 1713911460663,
    "last_updated": 1713911503961
},
{
    "workflowName": "appviewx_infra_cert_monitor_vw",
    "statusCode": 2,
    "toolTip": {},
    "requestId": "10",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Failed",
    "created_time": 1713900601119,
    "last_updated": 1713900602617
},
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
    "workflowName": "Citrix NetScaler Golden Config Compliance",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "9",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863547365,
    "last_updated": 1713863554960
},
{
    "workflowName": "F5 BIG-IP CVE Reporting",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "8",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863494143,
    "last_updated": 1713863503829
},
{
    "workflowName": "Notify Unused VIP-State|Status|Statistics",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "7",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863492551,
    "last_updated": 1713863506940
},
{
    "workflowName": "F5 BIG-IP Golden Config Compliance",
    "statusCode": 1,
    "toolTip": {},
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "requestId": "6",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863485796,
    "last_updated": 1713863499493
},
{
    "workflowName": "F5 and Citrix Golden Config Compliance Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "5",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863467761,
    "last_updated": 1713863548670
},
{
    "workflowName": "F5 BIG-IP CVE Reporting Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "4",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
    "created_time": 1713863467658,
    "last_updated": 1713863498618
},
{
    "workflowName": "Unused Objects Data Trigger",
    "statusCode": 1,
    "toolTip": {},
    "requestId": "3",
    "workorder_id": "0",
    "created_by": "system",
    "status": "Completed",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
            "created_time": 1713863467585,
            "last_updated": 1713863496543
        },
        {
            "workflowName": "Fetch F5 BIG-IP CVEs",
            "statusCode": 2,
            "toolTip": {},
            "requestId": "2",
            "workorder_id": "0",
            "created_by": "system",
            "status": "Failed",
            "created_time": 1713863467583,
            "last_updated": 1713863543217
        },
        {
            "workflowName": "appviewx_infra_cert_monitor_vw",
            "statusCode": 2,
            "toolTip": {},
            "requestId": "1",
            "workorder_id": "0",
            "created_by": "system",
            "status": "Failed",
            "created_time": 1713863467491,
            "last_updated": 1713863495223
        }
    ],
    "iTotalDisplayRecords": 19
},
"message": null,
"appStatusCode": null,
"tags": null,
"headers": null
}
```


Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Pause Request

Pause a particular workflow request.

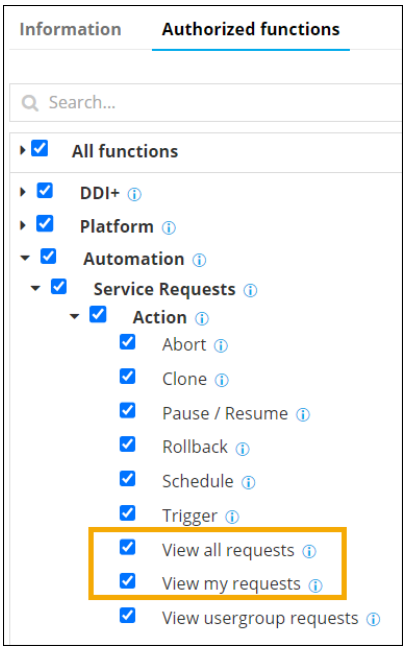
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**
- **Automation** > **Service Requests** > **Action** > **Pause/Resume**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/visualworkflow-pause-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-pause-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API; it is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username; it is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
	Example: "admin"
password *Header*	(Mandatory) AppViewX login password; it is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) Specifies the nature of the data in the payload. The parameter should be set to `application/json`. **Example:** "application/json"
gwkey *Query*	Tenant key; needed only in case of multi tenant installations.
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
requestId *Payload* *Query*	(Mandatory) Enter the request ID. **Type:** *String* **Example:** "ce7f1a14-2bf9-4e4a"

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	NA	Workflow Request Id is Mandatory. **Possible Remediation:** Check and ensure if a valid request id is given.
404 Not Found	NA	Request not found. **Possible Remediation:** Check and ensure if a valid request id is given.

Sample Request/Response

Use Case:

To pause a particular request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-pause-request?gwsource=external
```

Request Payload

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
NA
```

Sample Response

```
{
  "response": null,
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Resume Request

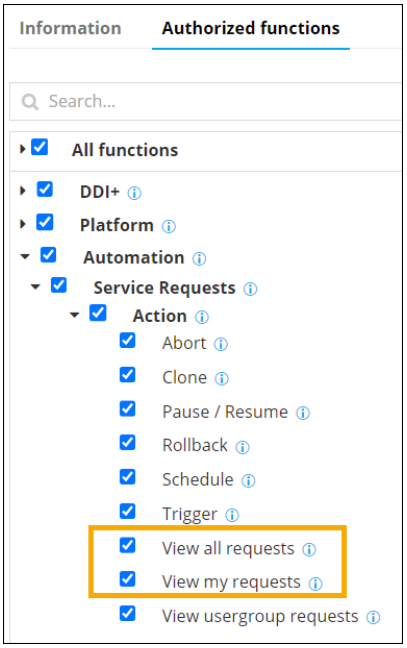
Resume a particular workflow request.

Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**
- **Automation** > **Service Requests** > **Action** > **Pause/Resume**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Request Structure

Endpoint:	/visualworkflow-resume-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-resume-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API; it is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username; it is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password; it is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) Specifies the nature of the data in the payload. The parameter should be set to `application/json`. **Example:** "application/json"
gwkey *Query*	Tenant key; needed only in case of multi tenant installations.
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
requestId *Payload* *Query*	(Mandatory) Enter the request ID. **Type:** *String* **Example:** "ce7f1a14-2bf9-4e4a"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	NA	Workflow Request Id is Mandatory. **Possible Remediation:** Check and ensure if a valid request id is given.
404 Not Found	NA	Request not found. **Possible Remediation:** Check and ensure if a valid request id is given.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response

Use Case:

To resume a particular request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-resume-request?gwsource=external
```

Request Payload

```
NA
```

Sample Response

```
{
  "response": null,
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Generate Commands

To generate commands for the command repository.

Request Structure

Endpoint:	`/generate-commands`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/generate-commands?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
payload *Body*	(Mandatory) Contains all the params to be sent in the request body for the post request. **Type:** *Map*

Response Structure

- **Status Code: 200 OK**
- **Message:** Success
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
500 Internal Server Error	AVX_INVALID_DEVICE	Device not present. **Possible Remediation:** Check whether the device given in payload is available in device inventory.
500 Internal Server Error	AVX_INVALID_REPO_COMMAND	No command present in repository. **Possible Remediation:** Check whether the provided intent is present in the command repository collection.
500 Internal Server Error	CMD_REPO_ERROR_0006	Proxy details not configured. **Possible Remediation:** Check whether proxy is configured in UI ,If proxy required toggle is set to true for that particular device.
500 Internal Server Error	CMD_REPO_ERROR_0007	Vendor specific details details are null ,hence not able to fetch the device credentials for AkamaiCPS device. **Possible Remediation:** Provide valid device details.
500 Internal Server Error	CMD_REPO_ERROR_0008	Access details are null ,hence not able to fetch the akamai device credentials.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
		Possible Remediation: Provide valid device details.

Sample Request/Response

Use Case:

To generate commands for the command repository.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/generate-commands?gwsource=external
```

Request Payload

```
{
   "payload": {
      "intent": "run_bash_command_test",
      "resource": "VW_F5V13_StandAlone",
      "attributes": {
         "command":"tmsh show sys license",
         "is_bash_shell": false
      }
   }
}
```

Sample Response

```
{
   "response": {
      "device": {
         "name": "VW_F5V13_StandAlone",
         "ip": "192.168.143.67",
         "deviceType": [
            "LTM",
            "GTM"
         ],
         "module": [
            "LTM",
            "BIG-IP DNS"
         ],
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
    "dataCenter": "absecon",
    "vendor": "F5",
    "version": "v13",
    "detailedVersion": "13.1.4 build 0.0.3",
    "subsystem": "LoadBalancer",
    "category": "ADC",
    "partition": [
        "TestNew",
        "SDET_Delete_LTM",
        "Other",
        "Common"
    ],
    "keywords": [
        "VW_F5V13_StandAlone",
        "192.168.143.67",
        "LTM",
        "BIG-IP DNS",
        "absecon",
        "F5",
        "v13",
        "13.1.4 build 0.0.3",
        "Manual Entry",
        "Managed",
        "Active",
        "422188ce-397a-4b35-012c4dcceb79",
        "BRACH-MISHE-BKWJH-AXYOQ-XRSNLES",
        "2022/04/11",
        "2022/05/13",
        "22",
        "TestNew",
        "SDET_Delete_LTM",
        "Other",
        "Common",
        "default",
        " ",
        "API"
    ],
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"access": [
    {
        "accessType": "API",
        "userName": "sdet_user_001",
        "password": "vault:v1:Ow0jvPeP5ujm1hFtnUxwPgWCmguOXgqznK/FZsFYtAuwO6V8",
        "key": "nb5sxl3080l59j9x88np9opif"
    }
],
"sameCredential": false,
"credentialType": "Manual Entry",
"credentialName": "",
"status": "Managed",
"registrationKey": "BRACH-MISHE-BKWJH-AXYOQ-XRSNLES",
"serialNumber": "422188ce-397a-4b35-012c4dcceb79",
"expiryDate": "2022/05/13",
"serviceCheck": "2022/04/11",
"manageCertificate": "Ignored",
"enableEvent": false,
"deviceCommunicationPort": {
    "sshPort": 22,
    "httpsPort": 443
},
"lastRebootTime": "11:34:49 up 38 days",
"haActive": "Active",
"syncGroup": "default",
"syncStatus": " ",
"syncFailoverGroupName": "",
"autoSyncEnabled": true,
"autoSyncNeeded": true,
"attributes": {
    "partition": [
        "all-partitions"
    ],
    "roles": [
        "admin"
    ],
    "user": [
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
            "sdet_user_001"
        ],
        "interfaceIp": [
            "192.168.40.4"
        ]
    },
    "devIndex": 0,
    "tmosVersion": "13.1.4-0.0.3",
    "createdBy": "admin",
    "noDetect": false,
    "createdDateTime": 1650627389747,
    "lastModifiedBy": "admin",
    "lastModifiedDateTime": 1650627681037,
    "hostName": "gs-f5-pe216.lab.appviewx.net",
    "previousDeviceStatus": "Managed",
    "deviceCommunication": "Success",
    "localLBConfigUpdated": "",
    "localLBConfigDownload": "",
    "localLBConfigParsing": "",
    "globalLBConfigUpdated": "",
    "globalLBConfigDownload": "",
    "globalLBConfigParsing": "",
    "communicationAddressType": "ip",
    "tokenBasedAuthentication": false,
    "lastConfigFetchTriggeredTime": 1650627435191,
    "managementIp": "192.168.143.67",
    "lastManagedTime": 1650627681011,
    "_id": "6262933df8341e46ae647cac",
    "vLans": {
        "vlan_RD_": [
            {
                "vlans": "vlan_RD_",
                "selfIps": [
                    "192.168.40.4"
                ],
                "subnetIp": [
                    "192.168.40.0/24"
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
                    ],
                    "rangeIps": [
                        "192.168.40.1 - 192.168.40.3",
                        "192.168.40.5 - 192.168.40.254"
                    ]
                }
            ]
        },
        "isAutoDetect": true,
        "isSecondaryCommunication": false,
        "isHostedOnHypervisor": false,
        "isPrimary": true
    },
    "command": [
        {
            "type": "command",
            "deviceName": "",
            "uniqueNumber": 1,
            "sequenceNumber": 0,
            "command": "run util bash -c \"tmsh show sys license\"",
            "newCommand": false,
            "status": "",
            "properties": {}
        }
    ]
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```


Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Generate Commands And Implement

To generate commands for the command repository and implementation.

Request Structure

Endpoint:	`/generate-commands-and-implement`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/generate-commands-and-implement?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
payload *Body*	(Mandatory) Contains all the params to be sent in the request body for the post request. **Type:** *Map*

Response Structure

- **Status Code: 200 OK**
- **Message:** Success
- **Headers:**
  - **Content-Type:** application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
500 Internal Server Error	AVX_INVALID_DEVICE	Device not present. **Possible Remediation:** Check whether the device given in payload is available in device inventory.
500 Internal Server Error	AVX_INVALID_REPO_COMMAND	No command present in repository. **Possible Remediation:** Check whether the provided intent is present in the command repository collection.
500 Internal Server Error	CMD_REPO_ERROR_0006	Proxy details not configured. **Possible Remediation:** Check whether proxy is configured in UI ,If proxy required toggle is set to true for that particular device.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Message and Remediation
500 Internal Server Error	CMD_REPO_ERROR_0007	Vendor specific details details are null ,hence not able to fetch the device credentials for AkamaiCPS device. **Possible Remediation:** Provide valid device details.
500 Internal Server Error	CMD_REPO_ERROR_0008	Access details are null ,hence not able to fetch the akamai device credentials. **Possible Remediation:** Provide valid device details.

Sample Request/Response

Use Case:

To generate commands for the command repository and its implementation.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/generate-commands-and-implement?gwsource=external
```

Request Payload

```
{
  "payload": {
    "intent": "common-avi-intent",
    "resource": "avi_v20",
    "attributes": {
      "input_dict": "{}",
      "request_type": "GET",
      "device_version": "20.1.3",
      "url": "https://192.168.142.203/api/virtualservice?name=test_saas_vs",
      "tenant_name": "admin"
    }
  }
}
```

Sample Response

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
   "response": {
      "status": "Success",
      "deviceName": "avi_v20",
      "logs": [
         {
            "type": "rest",
            "command": "https://192.168.142.203/api/virtualservice?name=test_saas_vs",
            "output": {
               "Transfer-Encoding": "chunked",
               "response_entity": "{\"count\": 1, \"results\": [{\"_last_modified\": \"1663667361860630\", \"active_standby_se_tag\":
\"ACTIVE_STANDBY_SE_1\", \"advertise_down_vs\": false, \"allow_invalid_client_cert\": false, \"analytics_profile_ref\":
\"https://192.168.142.203/api/analyticsprofile/analyticsprofile-305c2061-aa37-4b91-b5fd-ef1b77dfbf4c\", \"application_profile_ref\":
\"https://192.168.142.203/api/applicationprofile/applicationprofile-dbd5d9bd-7996-4b49-92ca-9f034324a0e0\", \"bulk_sync_kvcache\": false,
\"close_client_conn_on_config_update\": false, \"cloud_ref\": \"https://192.168.142.203/api/cloud/cloud-77c521f0-b1b6-45e2-a140-9eedc12189c1\",
\"cloud_type\": \"CLOUD_NONE\", \"delay_fairness\": false, \"east_west_placement\": false, \"enable_autogw\": true, \"enabled\": true, \"flow_dist\":
\"LOAD_AWARE\", \"flow_label_type\": \"NO_LABEL\", \"ign_pool_net_reach\": false, \"limit_doser\": false, \"marked_for_delete\": false, \"max_cps_per_client\":
0, \"name\": \"test_saas_vs\", \"network_profile_ref\": \"https://192.168.142.203/api/networkprofile/networkprofile-d1cf6c38-8428-4716-a170-f6e36dbd7938\",
\"pool_ref\": \"https://192.168.142.203/api/pool/pool-6ad1994a-1c5f-40ee-9546-9c2f0862d08e\", \"redis_db\": 1,
\"redis_port\": 5024, \"remove_listening_port_on_vs_down\": false, \"scaleout_ecmp\": false, \"se_group_ref\":
\"https://192.168.142.203/api/serviceenginegroup/serviceenginegroup-69cec0db-ac5a-4e0d-a464-6d11cde48da9\", \"self_se_election\":
true, \"services\": [{\"enable_http2\": false, \"enable_ssl\": false, \"port\": 80, \"port_range_end\": 80}], \"ssl_sess_cache_avg_size\": 1024,
\"tenant_ref\": \"https://192.168.142.203/api/tenant/admin\", \"tls_ticket_key\": [{\"aes_key\": \"Qpzy0H29GhmfyM6O6TUR0A==\", \"hmac_key\":
\"bUVgFuTX3l/Kv7Ufxrupwg==\", \"name\": \"4b253145-1058-40ae-9983-736daf1bc9c8\"}, {\"aes_key\": \"0TFukILKfmBFlx8aACcozw==\", \"hmac_key\":
\"ebGOv2ciOeWnE7/eV9AN6g==\", \"name\": \"384c4140-0e4e-4b00-a3a5-bd50628c255e\"}, {\"aes_key\": \"jF2dCsCAexOPSFg5hliwuA==\",
\"hmac_key\": \"qgOsgOYrOoTSsw8RFVJpwA==\", \"name\": \"dfbb4430-4b28-4832-b101-1e776074fcb7\"}], \"traffic_enabled\": true,
\"type\": \"VS_TYPE_NORMAL\", \"url\": \"https://192.168.142.203/api/virtualservice/virtualservice-136b7997-05e2-4e6b-9fd3-fd187b8270cf\",
\"use_bridge_ip_as_vip\": false, \"use_vip_as_snat\": false, \"uuid\": \"virtualservice-136b7997-05e2-4e6b-9fd3-fd187b8270cf\", \"version\": \"6\",
\"vh_type\": \"VS_TYPE_VH_SNI\", \"vip_runtime\": [{\"num_additional_se\": 0, \"requested_resource\": {\"num_se\": 1, \"num_standby_se\": 0},
\"vip_id\": \"0\"}], \"vrf_context_ref\": \"https://192.168.142.203/api/vrfcontext/vrfcontext-670d9c1d-b9a3-49c5-b593-089c70fc67c2\", \"vsvip_ref\":
\"https://192.168.142.203/api/vsvip/vsvip-995e5148-1161-4ae1-ac59-8a2e21d4dcb4\", \"weight\": 1}]}\n",
               "X-Content-Type-Options": "nosniff",
               "Access-Control-Allow-Methods": "GET",
               "Connection": "keep-alive",
               "sessionid": "sessionid=None; expires=Thu, 22-Sep-2022 06:51:35 GMT; HttpOnly; Max-Age=86400; Path=/; SameSite=None; Secure",
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
                "Access-Control-Allow-Headers":
 "Accept,Authorization,Cache-Control,Content-Type,DNT,If-Modified-Since,Keep-Alive,Origin,User-Agent,X-Mx-
ReqToken,X-Requested-With,X-Avi-Tenant,X-Avi-UserAgent,X-Avi-Tenant-UUID,X-CSRFToken,X-PORTAL-AUTHTOKEN",
                "Date": "Wed, 21 Sep 2022 06:51:35 GMT",
                "Allow": "OPTIONS, HEAD, GET, POST",
                "Referrer-Policy": "strict-origin",
                "X-Frame-Options": "SAMEORIGIN",
                "AVI_API_VERSION": "20.1.3",
                "Access-Control-Expose-Headers": "X-NEXT-CALL",
                "Strict-Transport-Security": "max-age=31536000; includeSubdomains",
                "Cache-Control": "private, no-cache, must-revalidate",
                "Access-Control-Allow-Credentials": "true",
                "Content-Security-Policy": "default-src 'self'; style-src 'self' 'unsafe-inline'; font-src 'self' data:; img-src 'self' data:;",
                "Vary": "Origin",
                "Set-Cookie": "sessionid=None; expires=Thu, 22-Sep-2022 06:51:35 GMT; HttpOnly; Max-Age=86400; Path=/; SameSite=None; Secure",
                "X-XSS-Protection": "1; mode=block",
                "status": "200",
                "Content-Type": "application/json"
            },
            "status": "Success",
            "timestamp": 1663743389000,
            "sequenceNumber": 0,
            "uniqueNumber": 1,
            "retriedOutput": [],
            "executionCount": 0
        }
    ],
    "checkCompletion": true,
    "autoSyncNeeded": false,
    "activeDeviceName": "avi_v20",
    "queueEnabled": false,
    "executeDevicesSequentially": false
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
```


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
  "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Rollback Request

To rollback a particular request.

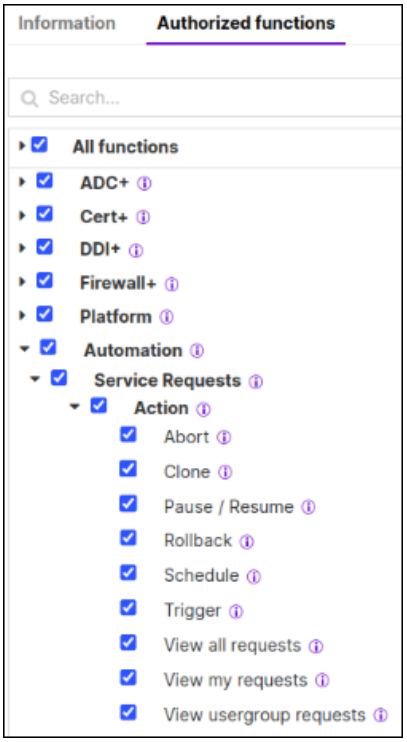
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Automation** > **Service Requests** > **Action** > **Rollback**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Request Structure

Endpoint:	/visualworkflow-rollback-request
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-rollback-request?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *Body*	(Mandatory) Contains all the params to be sent in the request body for the post request. **Type:** *map*

Response Structure

- **Status Code: 200 OK**
- **Message: Success**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	Contains the response object along with the hierarchy of the object mapped in field children.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
Key value pair	
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
400 Bad Request	avx-common-028	Invalid/ Incorrect payload. **Possible Remediation:** Check and ensure if a valid value is given in request payload field - input.
404 Not Found	engine-db-011	Request not found. **Possible Remediation:** Check and ensure if a valid requestId/workorderId is given.

Sample Request/Response

Use Case:

To rollback a particular request.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-rollback-request?gwsource=external
```

Request Payload

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
   "header": {
      "referenceRequestId": "41"
   },
   "data": {}
}
```

Sample Response

```
{
   "response": {
      "workorderId": "0",
      "requestType": "default",
      "requestId": "42",
      "message": "Workflow Request is created with Id 42 . Request submitted to workflow engine for processing workorder.",
      "status": "In Progress",
      "statusCode": 0
   }
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Execute Hook

It is used to execute a hook.

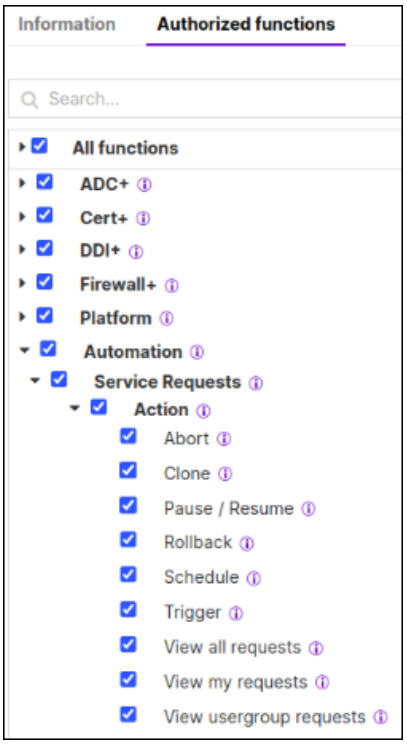
Before you begin

Before attempting to submit a request with this API, you must ensure that the following permissions are available:

- **Service Requests** > **Action** > **View all requests**
- **Service Requests** > **Action** > **View my requests**
- **Automation** > **Service Requests** > **Action** > **Trigger**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/execute-hook
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/execute-hook?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username	(Mandatory) AppViewX login username is required if sessionId is not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
Header	**Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *Body* *map*, *Object*	(Mandatory) Contains all the params to be sent in the request body for the post request. **Type:** *String*

Response Structure

- **Status Code: 200 OK**
- **Message:** Success
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message	Success message or failure description in case of error.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters (continued)

Name	Description
String	
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Success
500 Internal Error	NA	Invalid Hook Type. **Possible Remediation:**Check whether the given hook type is either api or script or query.
500 Internal Error	automation-hook-1012	Could not parse/convert Object to Json or Vice versa. **Possible Remediation:** Check whether the output is either an object or json.
500 Internal Error	automation-hook-1013	Error in rest communication. **Possible Remediation:** Check whether the details provided for rest call is valid.

Sample Request/Response

Use Case:

To execute a hook.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/execute-hook?gwsource=external
```

Request Payload

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
   "payload": {
      "hook": {
         "name": "Rest API"
      }
   }
}
```

Sample Response

```
{
   "response": {
      "status": "Success",
      "output": {
         "access_token": *******,
         "refresh_token": ******,
         "scope": "useraccount",
         "token_type": "Bearer",
         "expires_in": 1799
      },
      "type": "rest"
   },
   "message": null,
   "appStatusCode": null,
   "tags": null,
   "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Collection Operations

This section contains a collection of CRUD operations that supports add/modify/delete actions. Multiple documents can be added in a single request and it can be updated/deleted based on the query.

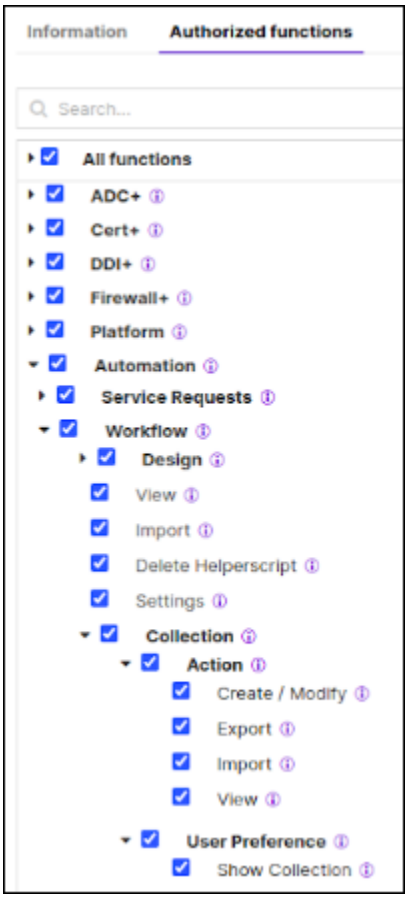
Before you begin

Before attempting to use collection operations, you must ensure that the following permissions are available:

- **Workflow** > **Collection** > **Action** > **Create/Modify**
- **Workflow** > **Collection** > **User Preference** > **Show Collection**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/collection-operations
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
Header	**Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
action *String*	(Mandatory) Action that has to be performed to the collection. Possible values are "add", "modify", and "delete". **Example:** "add"
query *JSON*	(Optional) Query to update/delete the collections document. (Mandatory if the action is “modify” or “delete”). **Example:** "*"deviceName": "F5"*"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
update *JSON*	(Mandatory) Fields to be updated/deleted in the collection (Mandatory if the action is "modify"). **Example:** *"fieldId": "device.connectionType",* *"value": "wlan0",* *"action": "overwrite"*
update.fieldId *String*	(Optional) Identifier key of the field that has to be updated. **Example:** "device.Bcast"
update.value *JSON*	(Optional) Values to be updated against the field identifier. **Example:** ""
update.action *String*	(Optional) Action to be performed on the field. Possible values are "overwrite", "append", and "remove". **Example:** "remove"
documents *JSON*	(Optional) Documents that have to be inserted to the collection. (Mandatory if the action is "add"). **Example:** ""
collection *String*	(Mandatory) Collection name that has to be updated. **Example:** "test"

Response Structure

- **Status Code: 200 Added/Updated/Deleted**
- **Message: Data added/Updated/Deleted successfully**
- **Headers:**
  - **Content-Type:** application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 Added/Updated/ Deleted	NA	Data added/Updated/Deleted successfully.
400 Bad Request	NA	The payload is not proper.
404 Not Found	NA	If the collection does not exist.

Sample Request/Response

Use Case: Add Document

Add the documents to the collections.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{
"payload": {
“collection”:”test”
"action": "add",
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"documents": [
{
"deviceName": "F5",
"ip": "1.2.3.4"
},
{
"deviceName": "A10",
"ip": "15.25.35.45"
}
]
}
}
```

Sample Response

```
{
"response": "Data added successfully",
"message": "Data added successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Sample Request/Response

Use Case: Delete Document

Delete the documents from the collections.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{
"payload": {
“collection”:”test”
"action": "delete",
"query": {
"deviceName": "F5"
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
}
}
```

Sample Response

```
{
"response": "Data deleted successfully",
"message": "Data deleted successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Sample Request/Response

Use Case: Update the Field (String or Numeric) Value

Modify the field (String or Numeric) value of the document.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{
"payload": {
"collection":"test"
"action": "modify",
"query": { "deviceName": "A10"
},
"update": [
{
"fieldId": "IP", "value": "4.5.6.7",
"action": "overwrite"
}
]
}
}
```

Sample Response

```
{
"response": "Data updated successfully",
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"message": "Data updated successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Sample Document

```
{
deviceName:"A10", IP:"1.2.3.4",
port:[99]
}
```

Updated Document

```
{
deviceName:"A10", IP:"4.5.6.7",
port:[99]
}
```

Sample Request/Response

Use Case: Append the Values into Existing Array Field

Append the values into an existing array field of a collection document.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{
"payload":
 “collection”:”test”
        "action": "modify",
        "query": { "deviceName": "A10"
},
"update": [
{
"fieldId": "port",
"value": [22,55,66],
"action": "append"
}
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
]

}

}
```

Sample Response

```
{

"response": "Data updated successfully",

"message": "Data updated successfully",

"appStatusCode": null,

"tags": null,

"headers": null

}
```

Sample Document

```
{

deviceName: "A10", IP: "1.2.3.4",

port: [99]

}
```

Updated Document

```
{

deviceName: "A10", IP:"4.5.6.7",

port:[99,22,55,66]

}
```

Sample Request/Response

Use Case: Modify the Nested Field Values

Modify the field values of a json object in a collection document.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{

"payload": {

“collection”:”test”

"action": "modify",

"query": { "deviceName": "F5"

},
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
"update": [
{
"fieldId": "device.connectionType",
"value": "wlan0",
"action": "overwrite"
}
]
}
}
```

Sample Response

```
{
"response": "Data updated successfully",
"message": "Data updated successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Sample Document

```
{
device: {
name: "F5",
ipV6: "fe80::e206:e6ff:fe23:e07/64",
connectionType: "eth0",
location: { dataCenter: "US",
clusterNodeId: "234rfg23456udcv"
}
}
}
```

Updated Document

```
{
device: {
name: "F5",
ipV6: "fe80::e206:e6ff:fe23:e07/64",
connectionType: "wlan0",
location: { dataCenter: "US",
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
clusterNodeId: "234rfg23456udcv"

}

}

}
```

Sample Request/Response

Use Case: Modify by Removing a Field from an Existing Document

Modifying a document by removing a field.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-operations?gwsource=external
```

Request Payload

```
{

"payload": {

“collection”:”test”

"action": "modify",

"query": { "deviceName": "F5"

},

"update": [

{

"fieldId": "device.Bcast",

"action": "remove"

}

]

}

}
```

Sample Response

```
{

"response": "Data updated successfully",

"message": "Data updated successfully",

"appStatusCode": null,

"tags": null,

"headers": null

}
```

Sample Document

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
{
device: {
name: "F5",
ipV6: "fe80::e206:e6ff:fe23:e07/64",
connectionType: "eth0",
location: { dataCenter: "US",
clusterNodeId: "234rfg23456udcv"
},
"Bcast": "192.168.138.255"
}
}
```

Updated Document

```
{
device: {
name: "F5",
ipV6: "fe80::e206:e6ff:fe23:e07/64",
connectionType: "wlan0",
location: { dataCenter: "US",
clusterNodeId: "234rfg23456udcv"
}
}
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Get Collection Details

API to get Collection details based on the query.

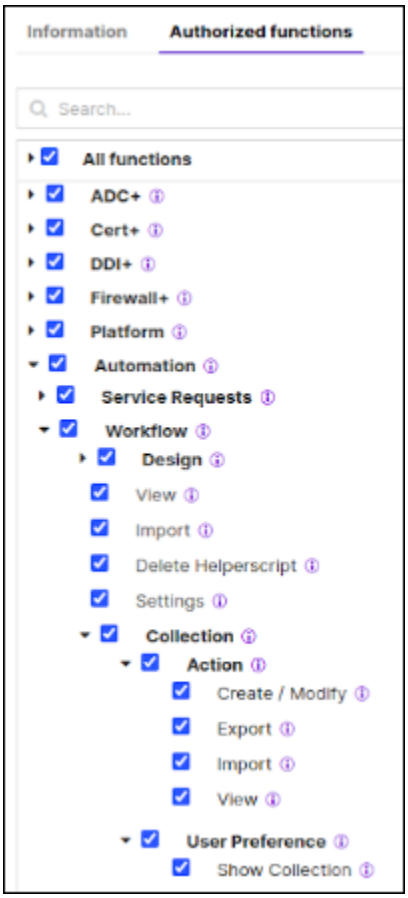
Before you begin

Before attempting to use collection operations, you must ensure that the following permissions are available:

- **Workflow** > **Collection** > **Action** > **View**
- **Workflow** > **Collection** > **User Preference** > **Show Collection**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Request Structure

Endpoint:	/collection-get-details
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-get-details?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionID	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
Header	**Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *String*	Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
startIndex *Payload*	(Optional) Start index of the data. Default value is 0. **Type:** *Nmber*
maxSize *Payload*	(Optional) Maximum number of data can be fetched. Default value is 100. **Type:** *Nmber*
queryParam.collectionName	(Mandatory) Collection name to get the details.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Name	Description
Payload	**Type:** *String*
filterValue *Payload*	(Optional) Filter based on keywords. **Type:** *String*

Response Structure

- **Status Code: 200 OK**
- **Message: Collection details fetched successfully**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Collection details fetched successfully.
400 Bad Request	NA	If the payload is not proper.(Invalid input).
404 Not Found	NA	If the collection does not exist.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Sample Request/Response

Use Case:

To get Collection details based on the query.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/collection-get-details?gwsource=external
```

Request Payload

```
{
  "payload": {
 "startIndex": 0,
 "maxSize": 5,
 "queryParam": {
   "collectionName": "ExcludeIP"
 },
 "filterValue": "172.168.23.25"
  }
}
```

Sample Response

```
{
  "response": {
 "data": [{
   "ip": "172.168.23.25",
   "uniqueKey": "ip",
   "orderBy": "ip",
   "_keywords": "[172.168.23.25, ip, ip, [172.168.23.25, ip, ip, Alaska, datacenter, General, ExcludeIP], Alaska, datacenter, General, ExcludeIP]",
   "datacenter": "Alaska",
   "_id": "5ec2c367d027f42e05c6e750",
   "groupBy": "datacenter",
   "type": "General",
   "collectionName": "ExcludeIP"
 }],
 "uniqueKey": "ip",
 "count": 1
  },
  "message": null,
  "appStatusCode": null,
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
 "tags": null,
 "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Collection Exists in DB

To find the collection details existing in the DB.

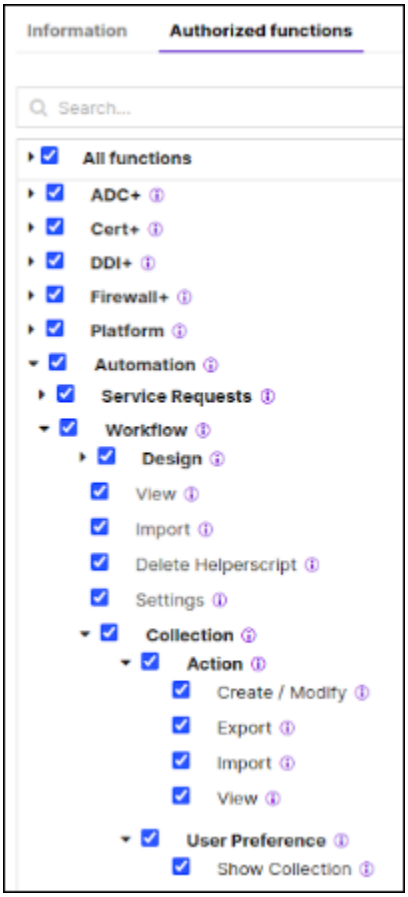
Before you begin

Before attempting to use collection operations, you must ensure that the following permissions are available:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Workflow** > **Collection** > **Action** > **View**
- **Workflow** > **Collection** > **User Preference** > **Show Collection**.

You can check this in the **Platform** module under **IDENTITY** > **Role** > **Authorized functions**.



Request Structure

Endpoint:	/aps-collection-exists
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/aps-collection-exists?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters

Name	Description
sessionID *Header*	(Mandatory) Use session ID retrieved from login API is required if username and password are not provided. **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"
username *Header*	(Mandatory) AppViewX login username is required if sessionId is not provided. **Example:** "admin"
password *Header*	(Mandatory) AppViewX login password is required if sessionId is not provided.. **Example:** "AppViewX@123"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external
Payload *String*	Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
collectionName *Payload*	(Mandatory) Collection name to get the details. **Type:** *String* **Example:** <collection_name>

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Structure

- **Status Code: 200 OK**
- **Message: Collection details fetched successfully**
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response *Key value pair*	Contains the response object along with the hierarchy of the object mapped in field children.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags *NA*	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Message and Remediation
200 OK	NA	Collection details fetched successfully.
400 Bad Request	NA	If the payload is not proper.(Invalid input).
404 Not Found	NA	If the collection does not exist.

Sample Request/Response

Use Case:

To find the collection details existing in the DB.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/aps-collection-exists?gwsource=external
```

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Request Payload

```
{
   "payload": {
               "collectionName": <collection_name>
      }
}
```

Sample Response

```
{
   "response": {
      "isExists": false
   },
   "message": null,
   "appStatusCode": null,
   "tags": null,
   "headers": null
}
```

Reference

Understanding the sample URL

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Update Workflow Resource Permission

This API is used to update resource permission for request execution, and user task-level RBAC of workflows. The user can assign and unassign resource permissions based on the action provided in the API payload. They can update the request resource permission of a workflow for any resource only if the user has workflow request permission.

Before you begin

- Session ID is obtained from the login API.
- Ensure that the session ID is valid and has not expired.

Request Structure

Endpoint:	`/visualworkflow-update-workflow-resource-permission`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-workflow-resource-permission?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
sessionId (or) Token *Header*	(Mandatory) Use either Session Id or Token received after login. **Type:** *String* **Example:** "sessionId": "ce7f1a14-2bf9-4e4a-89a8-bc780a255813"

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Input Parameters (continued)

Name	Description
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: **Type:** *String* • web • external
Payload *String*	(Mandatory) Input data for request body in application/json format. For payload details, see **Payload** section.

Payload

Name	Description
workflowNames *List<String>*	(Mandatory) Name of the workflow(s) to which the permission should be updated for resources. **Example:**["Update Resource Workflow"]
resources *List<String>*	(Mandatory) Resource name(s). **Example:**["ak_resource", "CLM Level1 Approver"]
action *String*	(Optional) assign/unassign. The default action will be “assign”. **Example:**assign

Response Structure

- **Status Code:** 202 Accepted
- **Message:** ACF assignment initiated for the given role.
- **Headers:**
  - **Content-Type:** application/json

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Response Parameters

Name	Description
response	Contains the response attributes for ACF assignment initiated for the given role.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Response Message
200 OK	null	{"SuccessWorkflows": ["<WorkflowName>"], "PartiallyUpdated or FailedWorkflows": ["<Failed workflow name if any>"] }
500	WORKFLOW_1000	Invalid input from the client. **Possible Remediation**: Provide a valid payload, or workflow name, or resource, or action value.
400 Bad Request	VALIDATION_ERROR_0006	Invalid '%s'::[Invalid '%s'. Field: workflowName]. **Possible Remediation**: Any one or few of the WorkflowName provided in the workflowNames array has invalid characters.
400 Bad Request	VALIDATION_ERROR_0006	Invalid '%s'::[Invalid '%s'. Field: resource]. **Possible Remediation**: Any one or few of the resource name provided in the resources array has invalid characters.
400 Bad Request	VALIDATION_ERROR_0008	%s' cannot exceed '%s' characters::['workflowName' cannot exceed '255' characters]

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

HTTP Code	appStatusCode	Response Message
		Possible Remediation: Any one or few of the WorkflowName provided in the workflowNames array has invalid length. Lenght must be between 2 to 255.
400 Bad Request	VALIDATION_ERROR_0008	'%s' cannot exceed '%s' characters::['resource' cannot exceed '128' characters] **Possible Remediation**: Any one or few of the resource name provided in the resources array has invalid length. Lenght must be between 2 to 128.
400 Bad Request	VALIDATION_ERROR_0009	Validation Failure: Field ${key} is empty::[Validation Failure: Field ${key} is empty. Field: resource] **Possible Remediation**: Any one or few of the resource name provided in the resources array is empty string or null.
400 Bad Request	VALIDATION_ERROR_0009	Validation Failure: Field ${key} is empty::[Validation Failure: Field ${key} is empty. Field: workflowName] **Possible Remediation**: Any one or few of the WorkflowName provided in the workflowNames array is empty string or null.

Sample Request/Response

Use Case

1. **Workflow Request ACL Permission Update**: Resource ACL permissions can now be assigned or unassigned for workflow requests.
2. **User Interface Palette ACL Permissions**:
   - RBAC permissions for user tasks can be assigned or unassigned at the task level within workflows.
   - For form tasks, **submit** permission will be provided during assignment.
   - For other user tasks, **read-write (RW)** permission will be granted.
3. **Audit Log Enhancements**: Audit logs will now include workflow names for which resource permissions were successfully updated, as well as those for which updates failed.

Sample Request

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/visualworkflow-update-workflow-resource-permission?gwsource=external
```

Request Payload

```
{
    "payload": {
  "workflowNames": [
    "Update Resource Workflow"
  ],
  "resources": [
    "ak_resource",
    "CLM Level1 Approver"
  ],
  "action":"assign/unassign"
}
}
```

Sample Response

```
 {
 "response": {
    "SuccessWorkflows": [
      "Update Resource Workflow"
    ],
     "PartiallyUpdated or FailedWorkflows": []
 }
 "message": "{SuccessWorkflows=[Update Resource Workflow], PartiallyUpdated or FailedWorkflows=[]}
 "appStatusCode": "WORKFLOW_1611",
 "tags": null,
 "headers": null
 }
```

Reference

Understanding the sample URL

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Chapter 3: Automation Troubleshooting Guide

This document helps you troubleshoot the common problems that you might encounter when using the functionalities in the Automation module.

• Prerequisites

• Issues with Workflows

• Issues with Workflow Tasks

• Issues in Using Helper Scripts

• Issues in Using Hooks

• Issues in Creating a Workflow Request

Prerequisites

Web Browser Requirement

Browser	Version	Notes
Firefox	Till latest (Version 84.0.4147.135)	NA
Chrome	Till latest (Version 80.0)	NA
IE	Limited support in 9, Full support from 10+	No support for IE9 post AppViewX Version 11.0
Safari	Till latest (Windows - Version 5.1.7, macOS - Version 13.1.2)	From AppViewX Version 11.1
Opera	Till latest (Version 70)	From AppViewX Version 11.1

Issues with Workflows

• Issues in Creating a Workflow

• Issues in Importing a Workflow

• Issues in Exporting a Workflow

• Issues in Workflow Validation

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Issues in Creating a Workflow

Following are the errors you may face when trying to create workflows, with their possible causes and solutions:

- Creating a new workflow or Renaming a workflow

Error Message	Possible Cause	Possible Solution
Workflow name already exists	Workflow name should be unique.	Enter some other name for the workflow.
Workflow name has invalid entry	Only a few special characters are allowed.	Allowed characters are '-' , '_' , '*' , ':' , '|' and space.

- Adding versions in a workflow

Error Message	Possible Cause	Possible Solution
Version limit exceeded	A workflow can have only 10 versions.	Create a workflow with 10 or less versions.

Issues in Importing a Workflow

Following are the errors you may face when trying to import workflows from other environment(s), with their possible causes and solutions:

- Workflow version already exists

Error Message	Possible Cause	Possible Solution
The workflow version already exists	Workflow is imported with the version which is already present.	Enter some other name for the version.

- Invalid workflow format

Error Message	Possible Cause	Possible Solution
Invalid workflow format	The file contains invalid entries and it cannot be parsed.	Choose an import file that has valid workflow details.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• Version limit exceeded

Error Message	Possible Cause	Possible Solution
Version limit exceeded	A workflow can have only 10 versions.	Check if the workflow has more than 10 versions and delete the versions that are not required.

Issues in Exporting a Workflow

Following are the errors you may face when trying to export workflows from AppViewX, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Cannot export more than 25 workflows at once	Only 25 workflows can be exported at a time.	Select only 25 or less workflows to export.

Issues in Workflow Validation

Following are the errors you may face when trying to validate workflows, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Invalid Workflow	There might be an issue in the workflow.	Avoid run time errors by validating the workflow.

• Illegal flow

Error Message	Possible Cause	Possible Solution
Invalid workflow - illegal flow	Multiple task outputs are given as an input to a single task.	Use Join task to provide multiple inputs to a task
Invalid workflow - illegal flow	A single task output is connected to multiple tasks.	Use Split task to connect single task output to multiple tasks.

• Broken flow

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Error Message	Possible Cause	Possible Solution
Invalid workflow - broken flow	A task is not connected to any other tasks in the workflow.	Check if all the tasks are connected.

• Subflow Validation

Error Message	Possible Cause	Possible Solution
Invalid workflow	There could be a validation error in the subflows.	Check if the subflow(s) have any illegal or broken flows.

Issues with Workflow Tasks

• Issues in Adding Tasks
• Issues in Updating Tasks
• Issues in User Interface Tasks

Issues in Adding Tasks

Following are the errors you may face when trying add tasks tasks to workflows, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Invalid Task component details	When an API is used to drag and drop a task, the payload might be wrong.	Drag and drop a task from another section such as User Interface or Notification and check for the payload.
Workflow not found	Workflow details might be missing in the database.	Check whether the workflow details exist in the database.
Workflow components detail not found	Workflow component details might be missing in the database.	Check whether the workflow details exist in the database.
Link Id is mandatory to perform the action	Task ID might be missing.	Check whether the task ID is available for the task.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Issues in Updating Tasks

Following are the errors you may face when trying update existing tasks in workflows, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Workflow Detail not found	Workflow details might be missing in the database	Check whether the workflow details exist in the database

Issues in User Interface Tasks

Following are the errors you may face when trying add User Interface tasks to your workflows, with their possible causes and solutions:

- Access Denied Issue in Form Task

Error Message	Possible Cause	Possible Solution
User is not authorized to perform the action	The logged in resource might not have access to the Form task.	1. Open the task and click **Resource & Settings.** 2. Under **Resource & Settings**, click **Request Resource**. 3. Give submitter access to the logged in resource.

- Access Denied Issue in Other User Interface Tasks

Error Message	Possible Cause	Possible Solution
User access denied	The logged in resource might not have access to the user interface task	1. Open the task and click **Resource & Settings.** 2. Under **Resource & Settings**, click **Request Resource**. 3. Give submitter access to the logged in resource.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

• Global Variable Issue in Form Task

Error Message	Possible Cause	Possible Solution
The referred global variable will be null	The global variable toggle might not be enabled when adding the field in the Form task.	Enable the global variable toggle in the Form task to refer it across the workflow.

Issues in Using Helper Scripts

• Issues in Creating a Helper Script
• Issues in Updating a Helper Script
• Issues in Importing a Helper Script

Issues in Creating a Helper Script

Following are the errors you may face when trying to create Helper Scripts, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Helper script name already exists	Helper Script name should be unique.	Enter some other name for the Helper Script.
Name has invalid entry	Only few special characters are allowed.	Allowed characters: '-' , '_' , '(', ')'"

Issues in Updating a Helper Script

Following are the errors you may face when trying to update Helper Scripts, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Syntax error/Indentation Error	Validated python script may contain syntax or indentation errors.	Fix the syntax or indentation errors in the python script.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Error Message	Possible Cause	Possible Solution
Module not found	Validated python script may contain modules that are not available in python/AppViewX or imported module name is wrong.	Import the module with the right name or add the required modules to AppViewX.

Issues in Importing a Helper Script

Following are the errors you may face when trying to import Helper Scripts, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Invalid helper script format	The file contains invalid entries and it cannot be passed.	Choose an import file that has valid helper script details.

Issues in Using Hooks

- Issues in Creating a Query explorer Hook
- Issues in Creating a Script Hook
- Issues in Creating a REST API Hook
- Issues in Exporting Hooks
- Issues in Importing Hooks

Issues in Creating a Query explorer Hook

Following are the errors you may face when trying to create hooks based on Query Explorer, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Hook name already exists in inventory	The Hook name should be unique.	Enter some other name for the Hook.
Please enter mandatory fields	Hook name/Projection fields may be missing.	Enter the Hook name and select the projection fields.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Issues in Creating a Script Hook

Following are the errors you may face when trying to create hooks based on Script, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Hook name already exists	The Hook name should be unique.	Enter some other name for the Hook.
Name has invalid entry	Only a few special characters are allowed.	Allowed characters: '-' , '_' , '(', ')'"
Syntax error/ Indentation Error	Validated python script may contain syntax or indentation errors.	Fix the syntax or indentation errors in the python script.
Module not found	Validated python script may contain modules that are not available in python/AppViewX or the imported module name is wrong.	Import the module with the right name or add the required modules to AppViewX.

Issues in Creating a REST API Hook

Following are the errors you may face when trying to create hooks based on REST API, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Hook name already exists	The Hook name should be unique.	Enter some other name for the Hook.
Name has invalid entry	Only a few special characters are allowed.	Allowed characters: '-' , '_' , '(', ')'"

Issues in Exporting Hooks

Following are the errors you may face when trying to export hooks from AppViewX, with their possible causes and solutions:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Error Message	Possible Cause	Possible Solution
Cannot export more than 25 hooks at once	Only 25 hooks can be exported at a time.	Select only 25 or less hooks to export.

Issues in Importing Hooks

Following are the errors you may face when trying to import hooks from other environments, with their possible causes and solutions:

Error Message	Possible Cause	Possible Solution
Hook already exists	The hook being imported is already available in the Hooks Inventory.	Enter some other name for the Hook.
Invalid hook format	The file contains invalid entries and it cannot be parsed.	Select an import file that has valid hook details.

Issues in Creating a Workflow Request

The **Service Requests** section in the **Automation** module gives you a single point view of all the assigned workflows, service requests triggered, and scheduled workflows. You can face the following issues when trying to trigger a workflow service request.

- Issues in Submitting a User Task
- Issues in Submitting an Executed Service Request
- Issues in Reflection of Workflow Data Updates in a Service Request
- Issues in the Form task in a Service Request
- Issues in Data Manipulation in a Service Request
- Issues in Command Task in a Service Request

Issues in Submitting a User Task

Tasks that display the data in a specific format and require a user's approval for further action are called user tasks. Following are the possible limitations that users might encounter during task execution:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- When an **unauthorised user** tries to submit a user task, unauthorized exception will be thrown.
- When a user task is the first task of a workflow and an unauthorised user tries to trigger that workflow, unauthorized exception will be thrown.
- When a user triggers a service request with following tasks as first task of the workflow, they will not be visible in the stage-wise view of the Service Request:
  - If
  - Switch and
  - Split task

  However, you can view them in the Activity log on the **Request :: All** page.

- Clean up flow cannot contain work order tasks in it

Issues in Submitting an Executed Service Request

- A rolled back request cannot be rolled back again or cloned
- A request which is in **In Progress** state cannot be cloned or rolled back. Only requests with status as Completed, Failed, or Partial can be cloned or rolled back.

Issues in Reflection of Workflow Data Updates in a Service Request

- Any changes made o the workflow after triggering a service request will not be reflected in that particular request.
- Cloning a request will only comprise the request's data from which it has been cloned and will not reflect any changes made to that workflow later.

Issues in the Form task in a Service Request

- **Auto fetch** of associate script output will not occur if the Form task is saved as draft and opened again. It will load the data only in case of a newly triggered Form
- When an **RGF flow** is enabled in a Form task, the request execution will take place in the prescribed order by the respective users who are allowed to create, review, or submit the task in order.
- **Pseudo form** is onl supported in following tasks:
  - Slack
  - REST
  - REST (I)
  - Schedule

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Script
- Email

Issues in Data Manipulation in a Service Request

- Any workflow containing **global key(s)** with dot (.) will not be allowed to create a request [Mongo behaviour].
- When the MongoDB document containing the request data exceeds **16mb in size** due to any uploaded file (or) any huge global variable data injection (or) heavy logs and so on, the request will hang.

Issues in Command Task in a Service Request

- Command Task **execution time** is based on the response time of the device in which the commands are being executed.
- If multiple commands are configured in Command Task, there is no possibility of selecting a particular command for execution; all commands are executed.
- Command Task **execution order** cannot be scheduled within the task for each command.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Chapter 4: Service Catalog User Guide

Welcome to the complete guide to getting started with the Service Catalog. This guide describes AppViewX's Page builder tool and gives you step by step instructions on creating custom pages, incorporating various page elements like HTML pages, reports, workflow catalogs, forms, tables etc to suit user-specific requirements.

- Module Overview
- Prerequisites
- Getting Started with the Service Catalog
- Accessing the Service Catalog
- Getting Started with OOB Service Catalog
- Customizing Prebuilt Service Catalogs
- Designing a Custom Service Catalog
- Version Control
- Preview a Service Catalog
- Publishing a Service Catalog
- Sharing a Service Catalog
- Service Catalogs with Birthright Role
- Cloning a Service Catalog
- Setting up the Landing Page
- Hiding the Page Header
- Troubleshooting

Module Overview

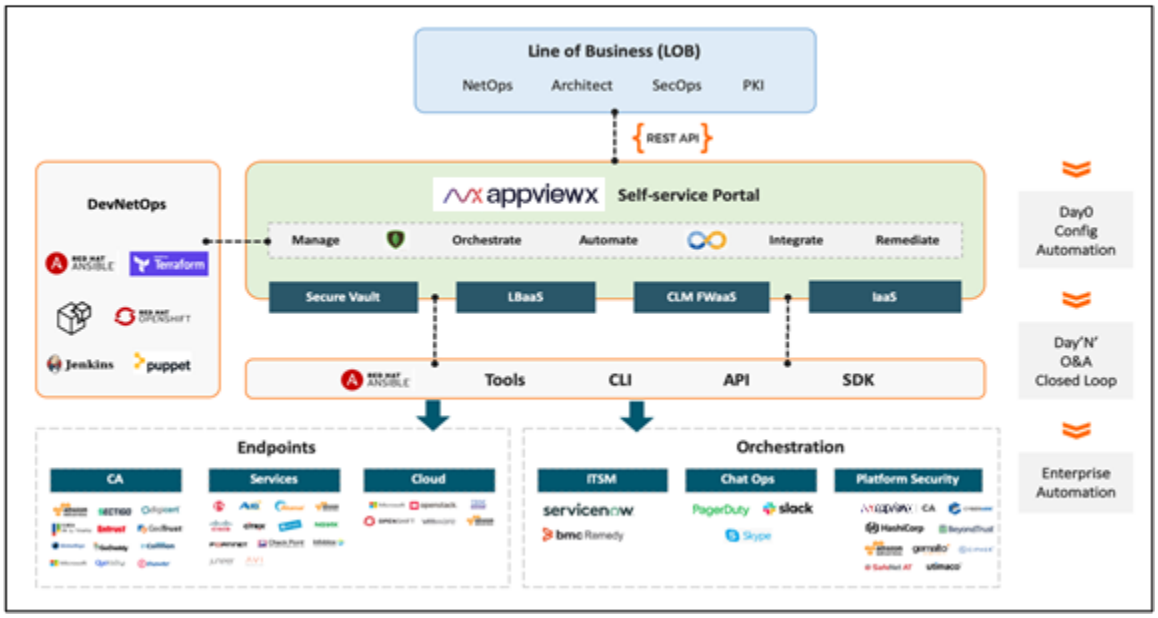
AppViewX's Service Catalog module allows you to design and customize your own catalog and access different modules within AppViewX from that catalog. You can design custom catalogs and add elements like HTML pages, quick links, buttons, search widgets and so on to suit specific requirements. Easy and quick access to multiple modules within one catalog saves time.

AppViewX's Service Catalog offers:

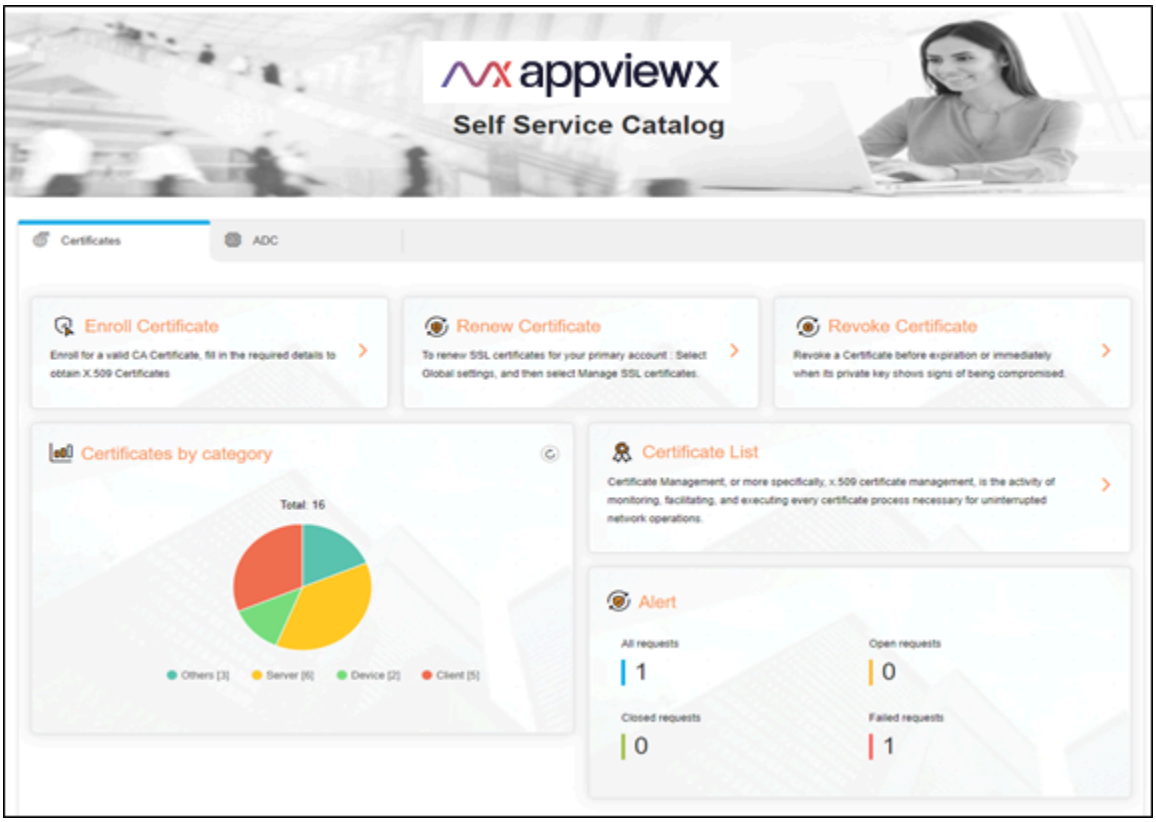
- Intuitive service catalogs to access service offerings for Line of Business (LOB), employees, and partners via customized pages.
- Service catalogs by persona for NetOps, PKI, Application Teams, SecOps.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Low code page builder that allows customization of pages with personalized task views.
- Provision to design custom web pages using HTML.
- Configurable page elements and customised branding to provide a seamless intuitive experience for end users.
- Provision to share catalogs for collaboration between team members.
- Reusability of page components among multiple users.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Prerequisites

The avx_visual_page_builder plugin must be enabled during AppViewX deployment.

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

Getting Started with the Service Catalog

AppViewX's Service Catalog is a self-service tool that allows you to:

- Use existing catalogs for immediate use
- Modify existing catalogs according to user-specific needs

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Design and customize catalogs by persona
- Assign permission for users to view, modify and share catalogs
- Publish self service catalogs.

Accessing the Service Catalog

New Menu	Old Menu
From the main navigation menu, select **Automation**. The **Automation** module is displayed with the **SERVICE CATALOG** accessible from the navigation pane on the left.	From the main navigation menu, select **Pages**. The **Pages** module is displayed with the **Published** section open by default.
Note: For more information on how to switch between menus, click here.	

The Service Catalog module has the following sections:

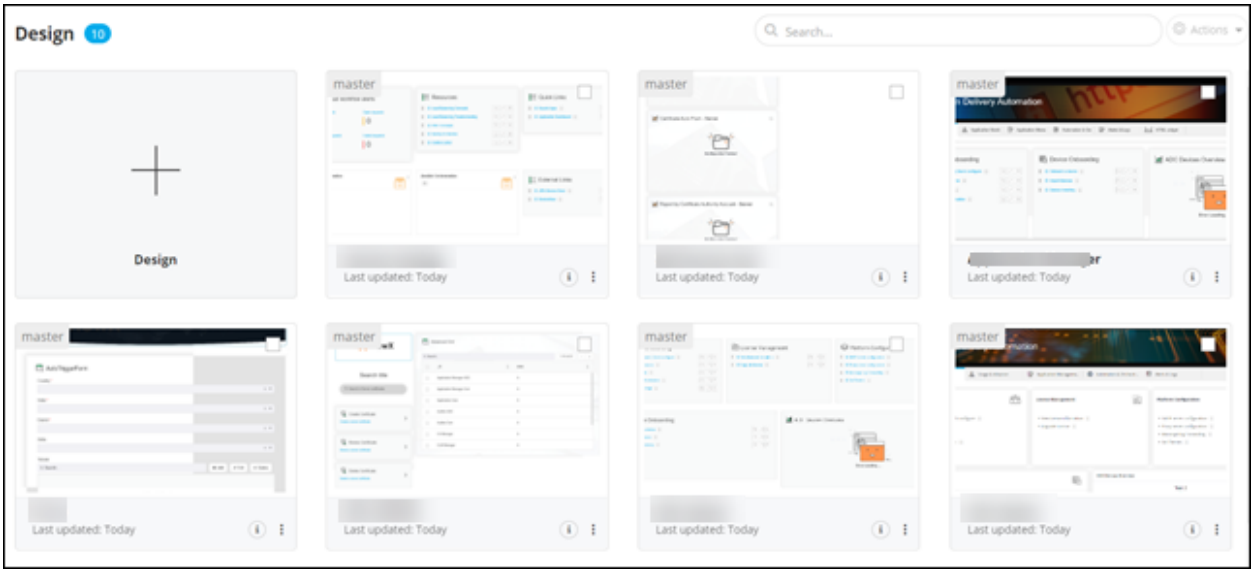
- **Published**: This is an inventory of all published catalogs.
- **Design**: This is an inventory of all catalogs in design mode.
- **Store**: This is an inventory of out of the box (OOB) catalogs.
- **Import**: This section allows you to import catalogs from other environments.

- Design
- Published
- Store
- Import

Design

The **Design** page shows all catalogs created by a user that are in design mode and have not been published yet.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



To access the inventory of catalogs in design mode:

New Menu	Old Menu
1. From the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, under **SERVICE CATALOG**, click **Design**.	1. From the main navigation menu, select **Pages**. 2. In the **Pages** module, from the left menu, click **Design**.
Note: For more information on how to switch between menus, click here.	

Various options in design section & their descriptions

Options	Description
	Displays ‘Last Updated’ info and description of the page.
	Allows users to perform the following actions on the selected page: • **Clone** • **Delete** • **Export**.

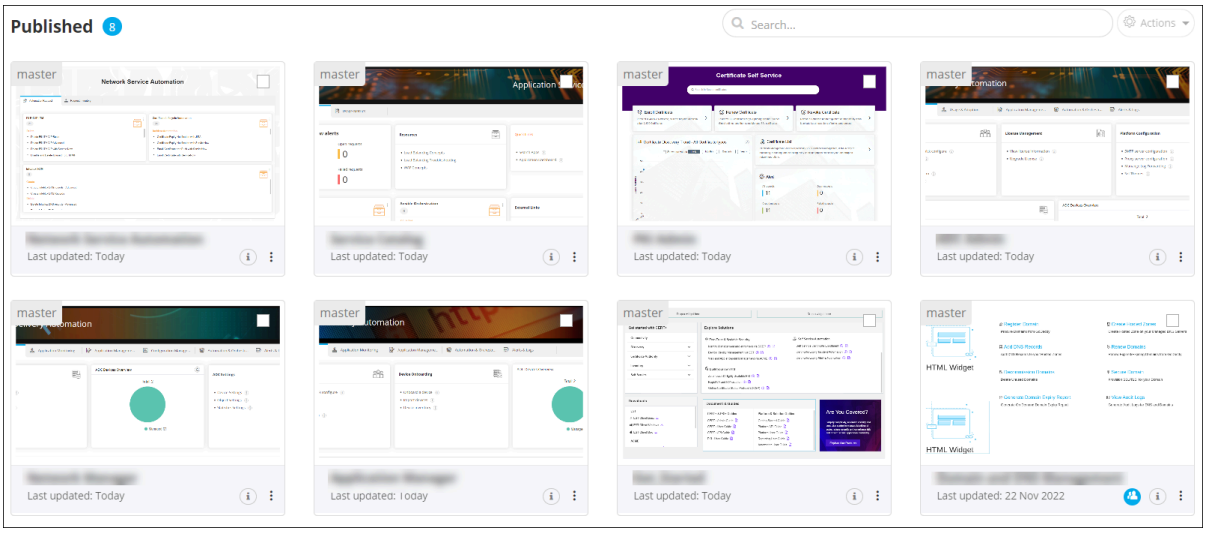
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Various options in design section & their descriptions (continued)

Options	Description
Search bar	Allows users to search for a specific page within the Design Inventory by using keyword(s).
Actions	Allows users to perform the following actions on selected page(s): • **Delete** • **Export**. **Note:** This option is enabled only when you select a page(s).

Published

This **Published** page shows all the catalogs that are ready-to-use and published.



To access the inventory of published catalogs:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

New Menu	Old Menu
1. From the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, under **SERVICE CATALOG**, click **Published**.	From the main navigation menu, select **Pages**. The **Pages** module is displayed with the **Published** section open by default.
Note: For more information on how to switch between menus, click here.	

Various options in published section & their descriptions

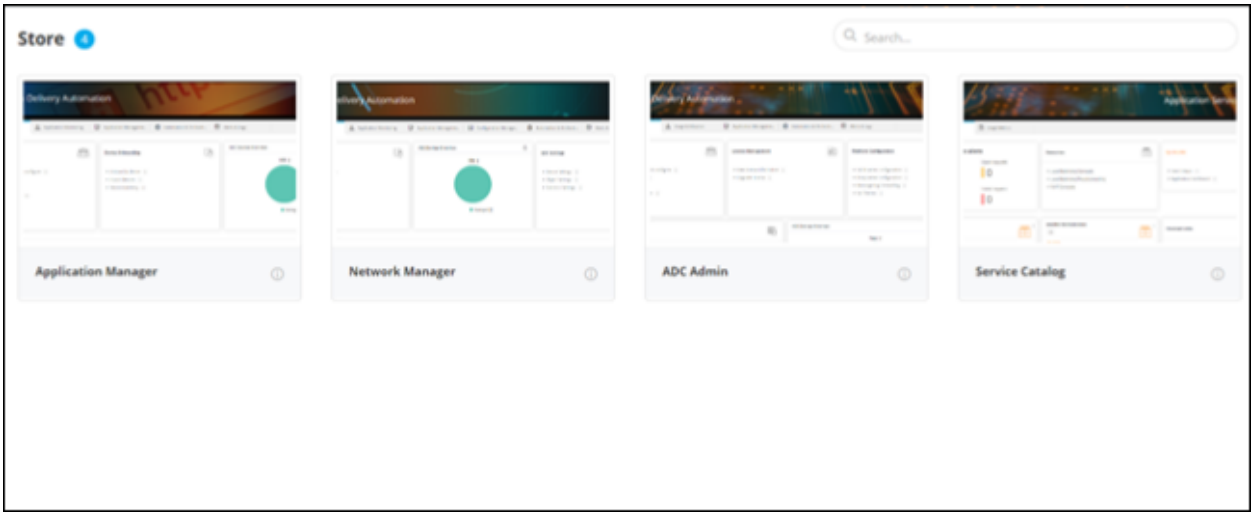
Options	Description
	A highlighted icon on the bottom right corner of the card represents a shared page. Hovering over the icon displays the name of the person who created the page.
	Shows ‘Last Updated’ info and description of the page.
	Allows users to perform the following actions on the selected page: • **Clone**: Allows users to clone the selected page. • **Export**: Allows users to export the selected page.
Actions	Allows users to export the selected page(s). **Note:** This option is enabled only when you select a page.
Search bar	Allows users to search for a specific page within the Published inventory by using keyword(s).

Important: Published catalogs cannot be deleted.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Store

Store has a collection of prebuilt out of the box (OOB) catalogs that can be used as a reference for building your customized catalogs. Select the OOB Catalog you want to build based on the persona and customize it as per your preference.



To access the inventory of OOB catalogs:

New Menu	Old Menu
1. From the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, under **SERVICE CATALOG**, click **Store**.	1. From the main navigation menu, select **Pages**. 2. In the **Pages** module, from the left menu, click **Store**.
Note: For more information on how to switch between menus, click here.	

Note: For more information on Getting Started with OOB Catalogs, click here.

Import

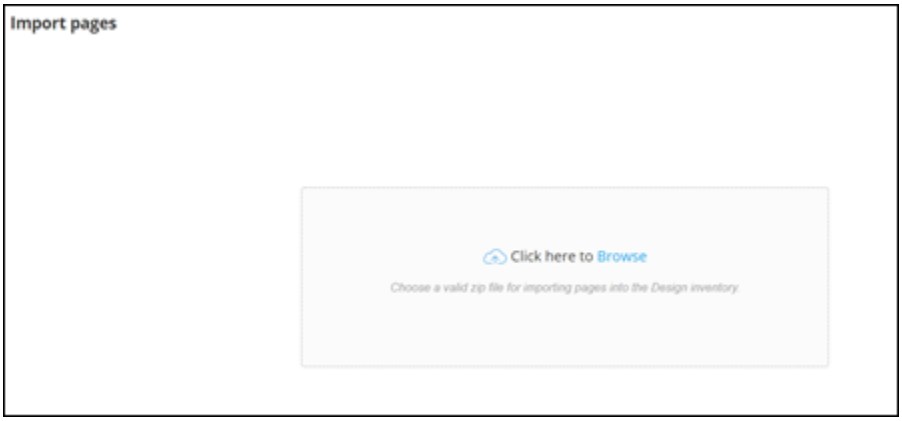
You can import a page into the AppViewX Service Catalog. Importing catalogs allows you to reuse catalogs exported from one environment into another.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

To import a catalog:

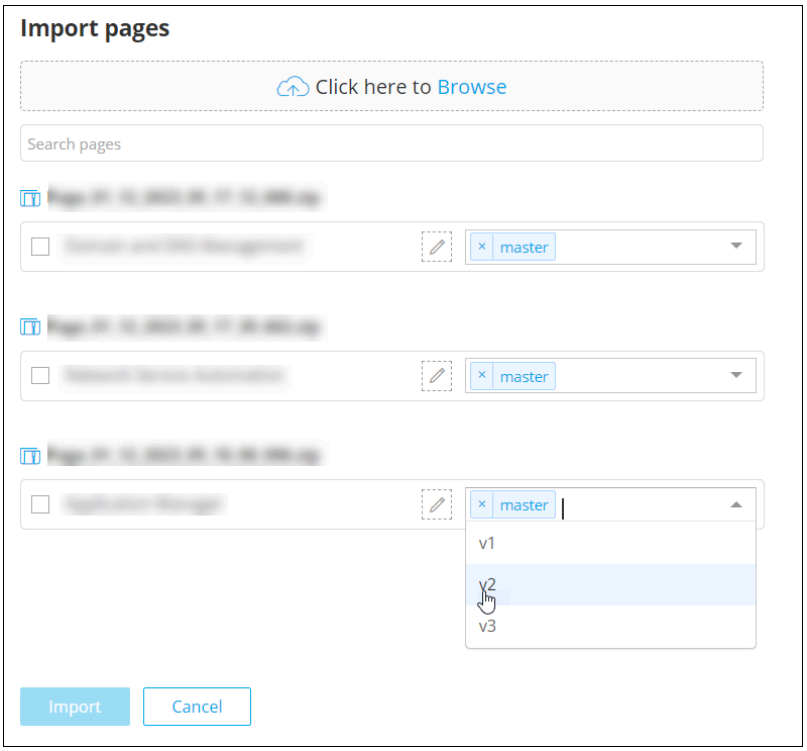
New Menu	Old Menu
1. From the main navigation menu, select **Automation**. 2. In the **Automation** module, from the left menu, under **SERVICE CATALOG**, click **Import**.	1. From the main navigation menu, select **Pages**. 2. In the **Pages** module, from the left menu, click **Import**.
Note: For more information on how to switch between menus, click here.	

1. To select a file to upload from your machine, click **Browse**.



2. Select the file(s) to be uploaded.
3. Once the files are uploaded, select the version for each catalog.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. To change the name of the catalogs before importing it into the Design inventory, click **(Edit)** icon next to the name and type in the new name.

5. Once you have selected the required version, click **Import**.
   The imported catalog is added to the Design Inventory.

Getting Started with OOB Service Catalog

Before you begin designing your self-service catalogs, here's a brief overview of getting started with pre-built catalogs. These prebuilt catalogs give you an insight into the various actions that each persona (such as a Network Manager, ADC admin and so on) would be able to perform using their customized catalog.

These prebuilt catalogs enable you to perform the following key actions:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

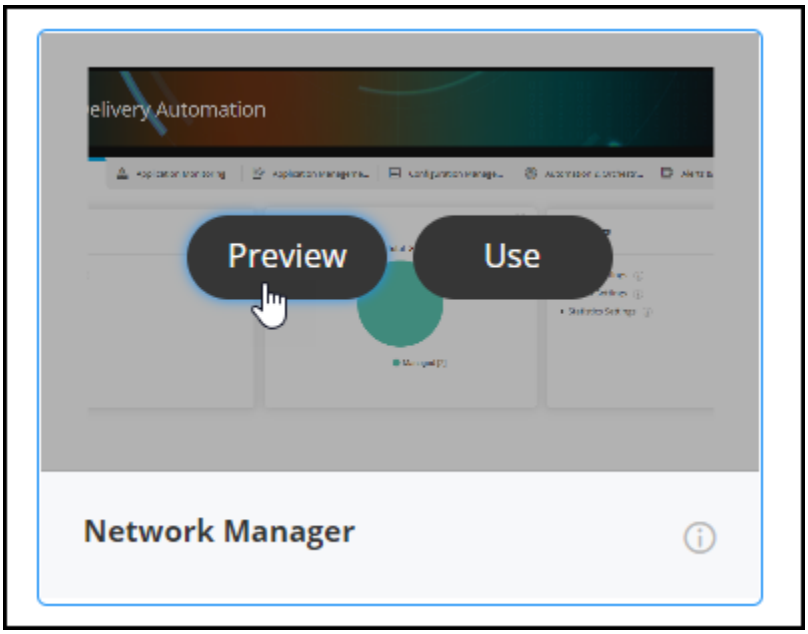
- Gain insights and visibility into application delivery services.
- Monitor and troubleshoot application services.
- Configuration management and upgrades.
- Self service and automation of application delivery services.

Based on the persona, be it a CA manager, ADC admin or an Application Manager, you can customize these catalogs and modify components to meet your requirements.

- Using OOB Catalogs
- Take a Tour

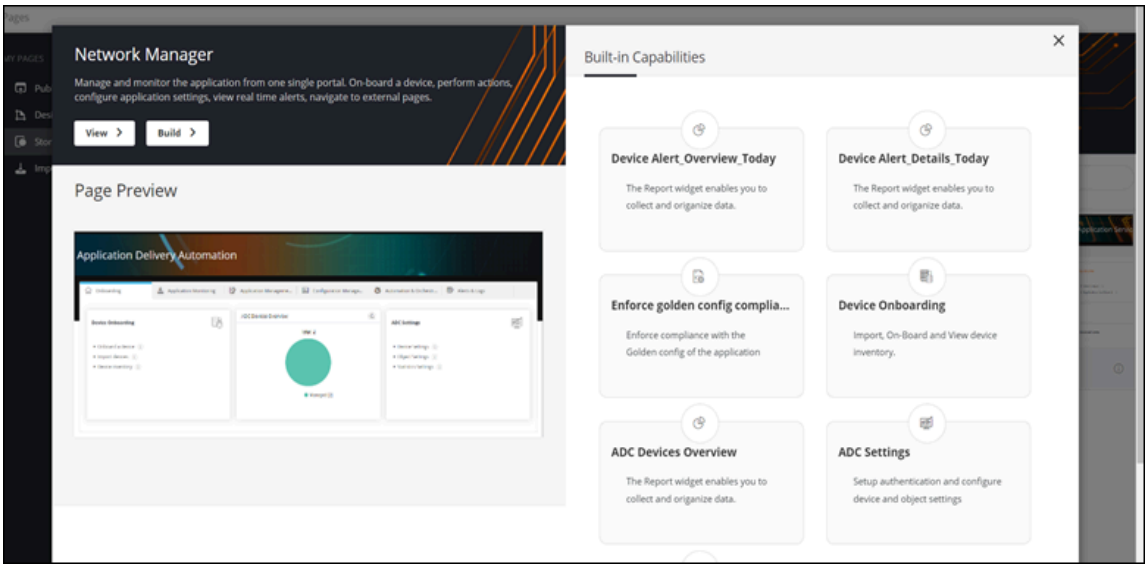
Using OOB Catalogs

1. Go to the Store.
2. To preview a prebuilt OOB catalog, hover your mouse over a catalog and click **Preview**. For example, **Network Manager**.



A preview of the Network Manager's page is displayed in a pop-up window.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



3. To view the complete catalog, click **View**.

   The catalog opens in a new tab.
4. To modify this catalog, on the pop-up page, click **Build**.

   A copy of this catalog is added to the **Design** inventory and the page opens in the same tab.

   You now have the option to:

   - Publish and use this OOB catalog as is.
   - Customize this catalog to suit persona-specific needs.
   - Design a new custom page from scratch.

Take a Tour

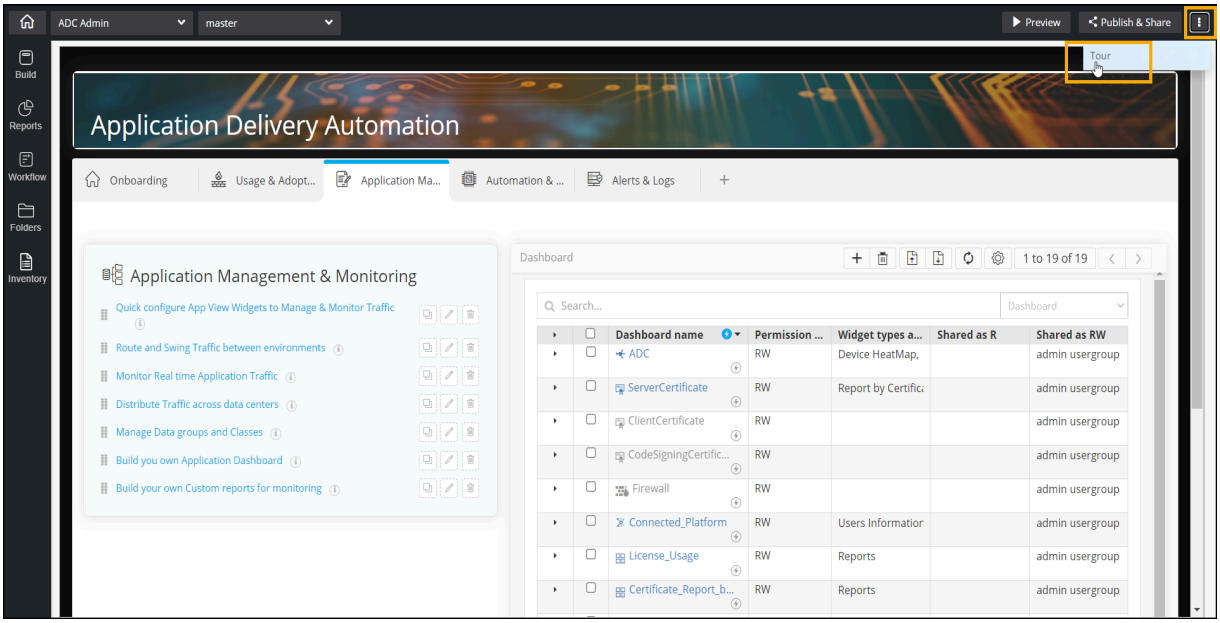
Take a quick tour of the Service Catalog module and familiarize yourself with its various functionalities. Users will be prompted to take a tour when they access the Service Catalog and build a custom catalog for the first time. There is also a provision to explore the tool from any catalog in design mode.

To take a tour:

1. Open a page from the Design section.
2. From the top right corner of the screen, click .

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. To explore the Service Catalog module, click **Tour**.



Customizing Prebuilt Service Catalogs

The Service Catalog module allows you to preview out of the box (OOB) catalogs and customize them according to their specific requirements.

1. Navigate to the Design inventory and click **Design**.
2. In the **Design page** window, to customize an existing page, hover your mouse over the page and click **Use**.

   A copy of the selected page opens on your screen and is added to the Design inventory.

> **Note:** For steps on designing catalogs, click here.

> **Tip:** You can also access and customize these catalogs from the Store.

Designing a Custom Service Catalog

- Getting Started with Designing a Custom Catalog
- Page Layout
- Build
- Reports

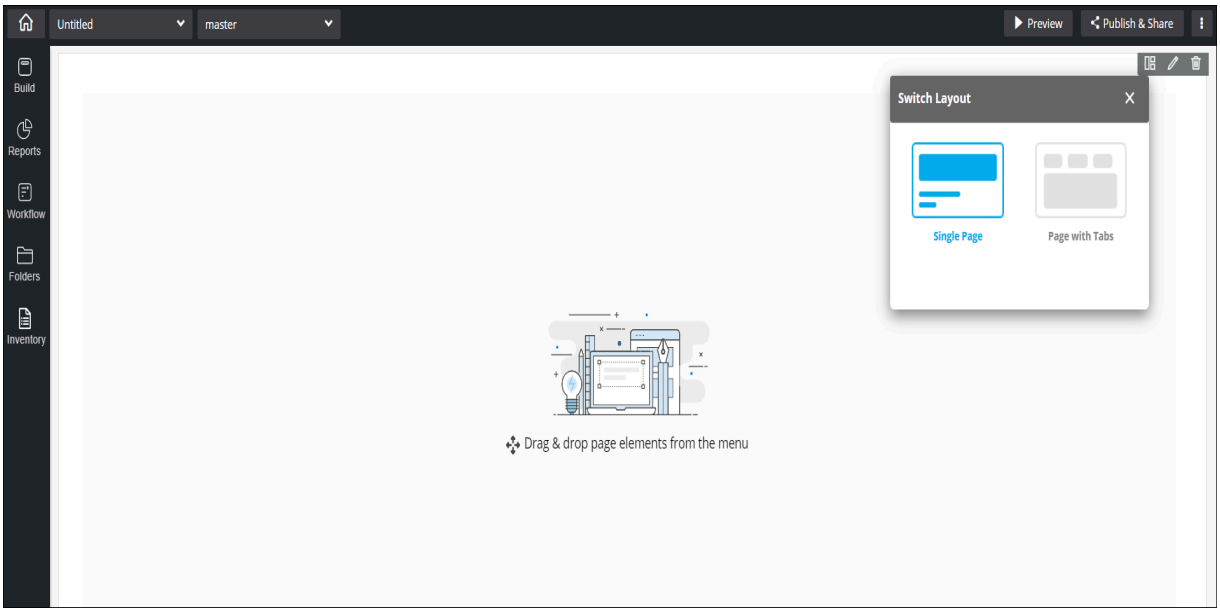
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Workflow
- Folders
- Inventory
- How to Undo/Redo actions
- Moving widgets
- Cloning Widgets

Getting Started with Designing a Custom Catalog

1. Go to the Design inventory and click **Design**.
2. In the **Design page** window, click **New Page**.

   A blank page is displayed with the **Switch Layout** window open by default.



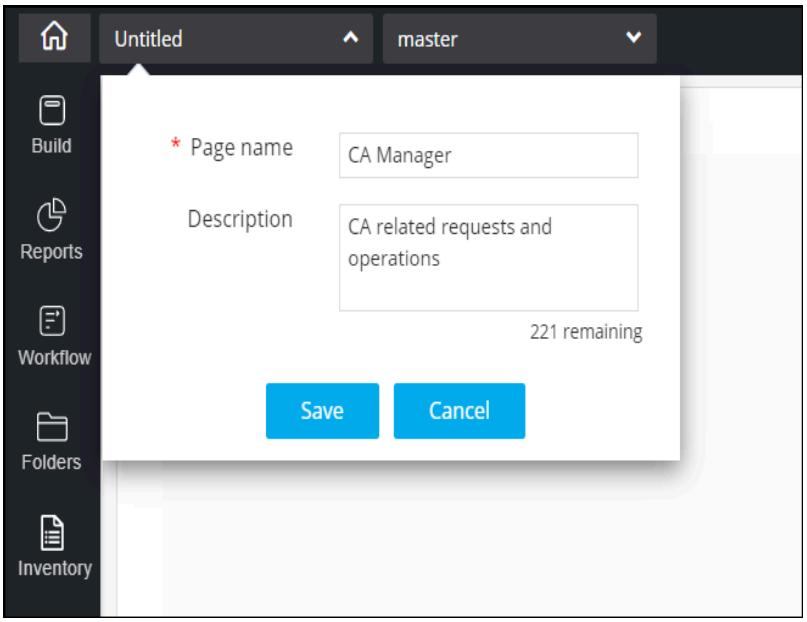
Note: For more information on page layouts, click here.

The component panel on the left shows the following options for Page elements:

- **Build**: Allows you to add widgets and form components to your catalog.
- **Reports**: Allows you to add report widgets to your catalog.
- **Workflow**: Allows you to add workflow catalogs to your catalog.
- **Folders**: Allows you to save catalogs to personal and shared folders.
- **Inventory**: Allows you to access OOB forms and customized forms.

3. To add a **Page name** and **Description,** click on the **Untitled** dropdown.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: Page name is a mandatory field.

4. Click **Save**.

Page Layout

You can choose between two layouts for your customized catalog:

- **Single page**: This is the default layout.
- **Multi Page**: You can create catalogs with up to 6 **Tabs** or create a page using **Stepper**.

- Switching from Single Page Layout to Page with tabs Layout
- Switching from Page with tabs Layout to Single Page Layout
- Tab reordering
- Modifying page layout
- Deleting a Catalog

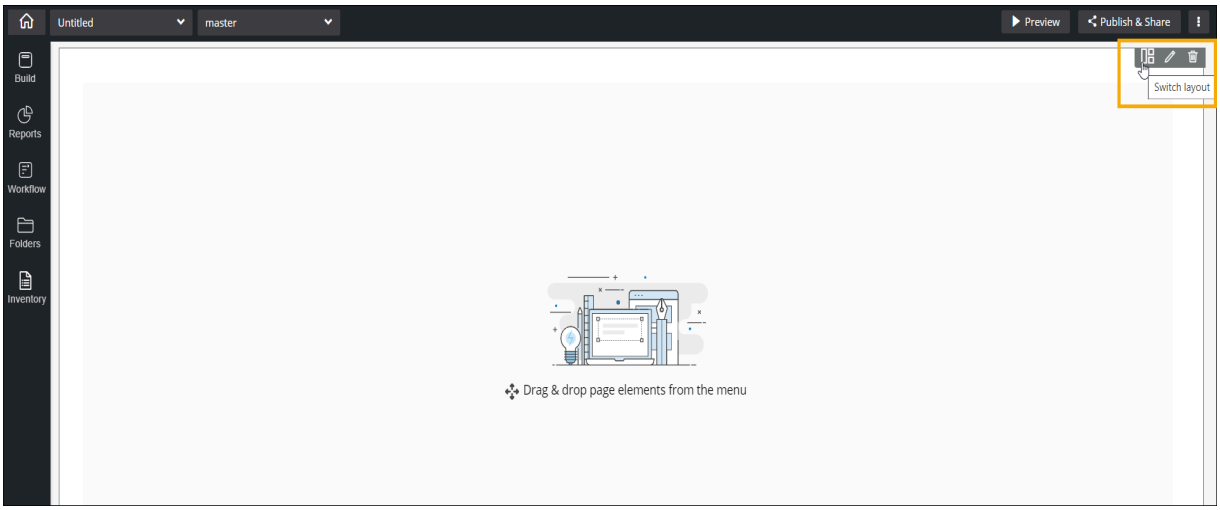
Switching from Single Page Layout to Page with tabs Layout

You can create a customized catalog with multiple tabs. There is a provision of adding up to 6 tabs in a catalog. The tabs can be arranged either horizontally or vertically.

To switch from single page layout to page with tabs/step layout:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. In the build area, from the top right corner of the screen, click **(Switch Layout)** icon.



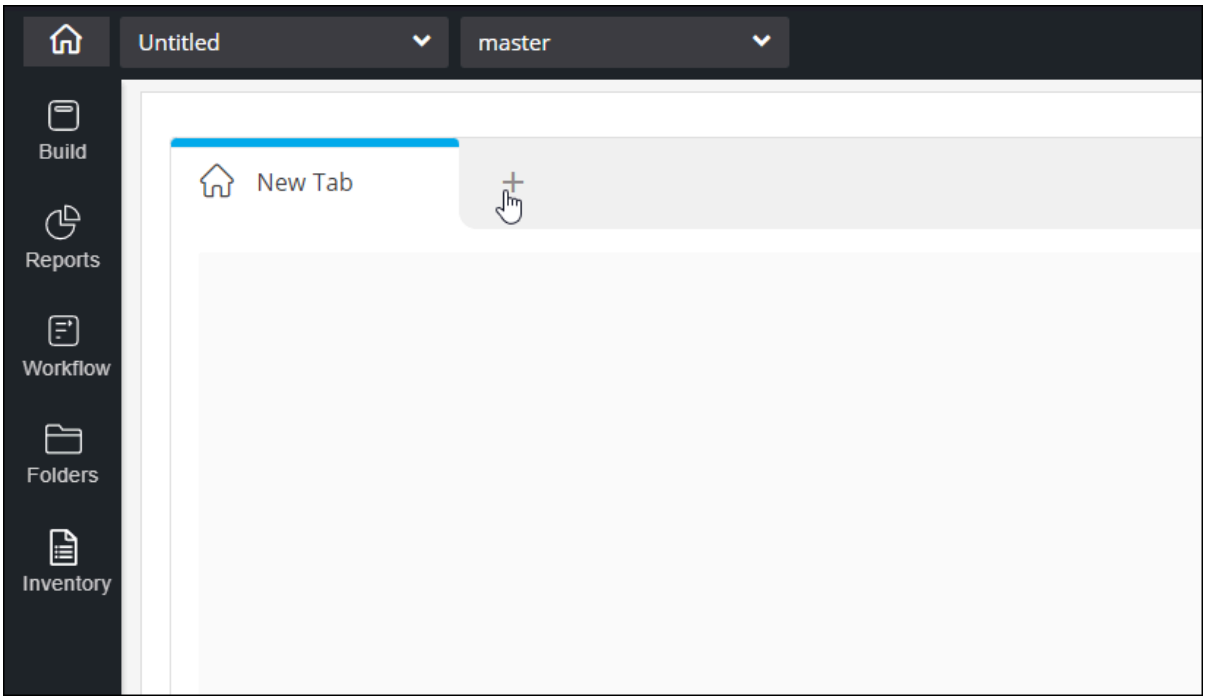
2. In the **Switch Layout** window that is displayed, click **Multi Page**.
3. Click **Yes** in the **Confirmation** pop-up window.

   The Tab setup page will be displayed by default..

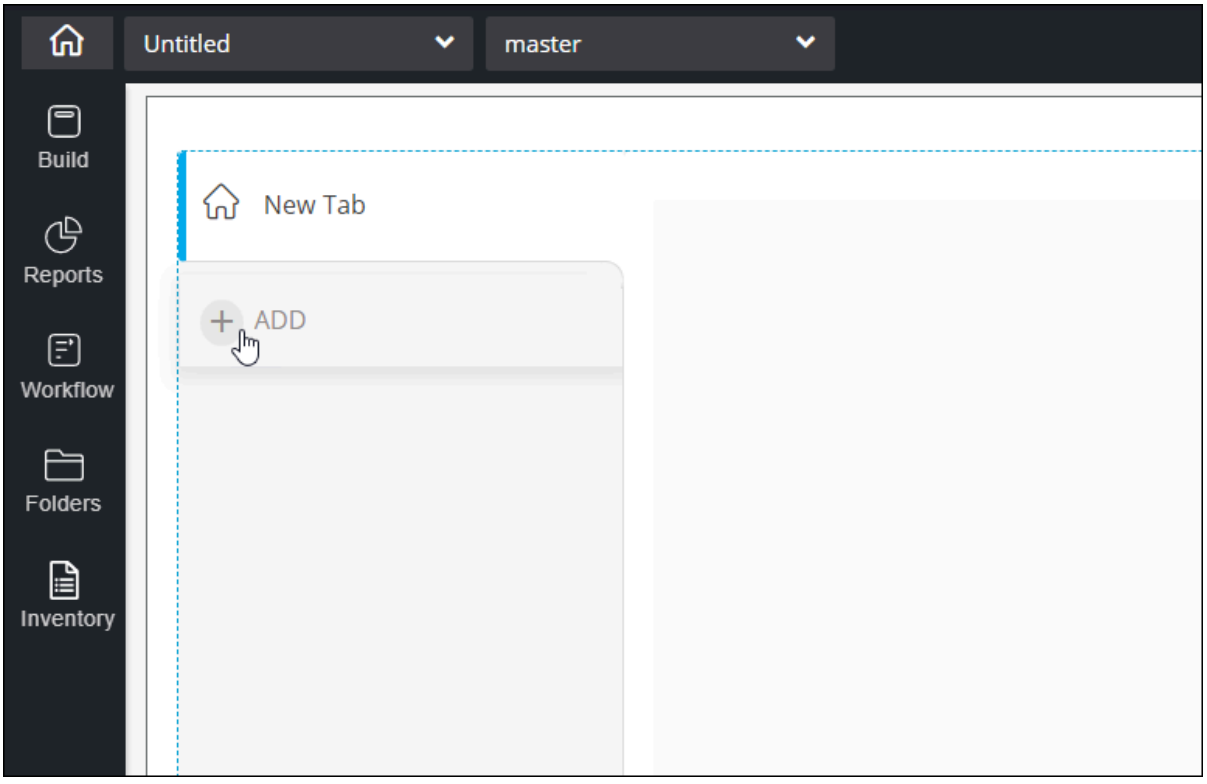
Multi Page with Tabs

1. To flip between horizontal and vertical tabs, from the top right corner of the screen, click **(Flip Right)** icon.
2. To add a tab, when tabs are horizontally oriented, click **(Add)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

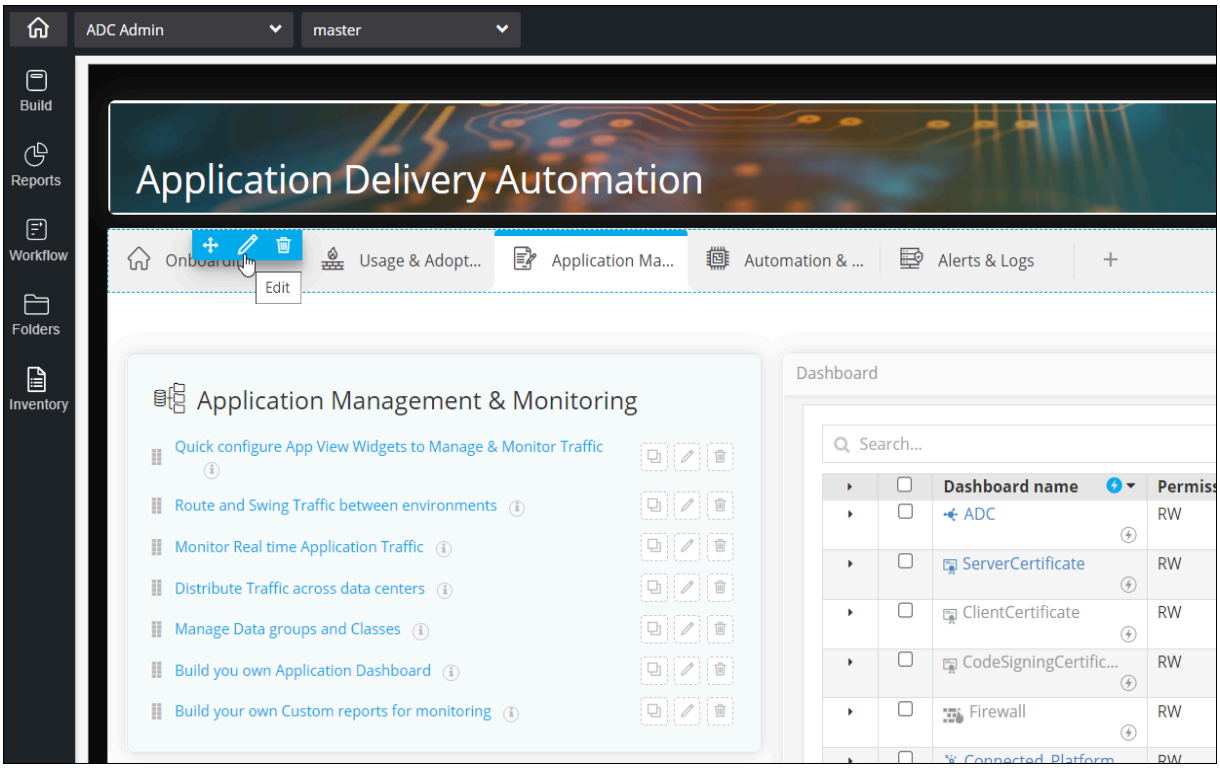


3. To add a tab, when tabs are vertically oriented, click **ADD**.



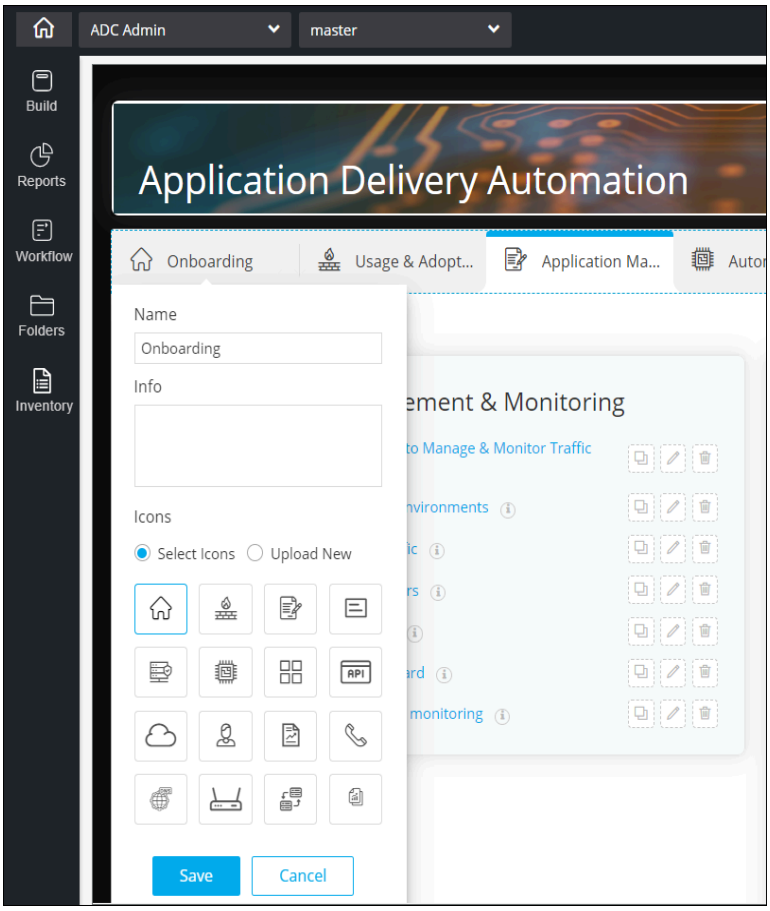
4. To modify the tab, hover your mouse over the tab and click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



A pop-up window is displayed right below the tab.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The following options are available here:

- **Name** - Allows you to change the name of the tab.
- **Info** - Allows you to add a tool-tip for the tab.
- **Icons** - Allows you to select an icon for the tab from the options displayed or upload a new icon.

5. Click **Save**.
6. To delete a tab, hover your mouse over the tab and click **(Delete)** icon.

Multi Page with Stepper

1. To switch to the Stepper layout click the (Edit) icon on the right.
   The **Layout** pop-up is displayed with **Properties** tab.
2. In the Select Layout Type section, click **STEPPER** and click **Save**.
   A panel with **Step 1** is displayed on the left.

Note: Once the STEPPER mode is save, you will not be able to move back to the Tab mode.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Click the (Edit) icon on the panel.
   The Stepper pop-up is displayed with **Properties** tab for the stepper configurations.
4. To change the 'Step 1' label, in the **Properties** > **Stepper Configurations** section, click the arrow icon.
   The details screen is displayed.
5. Enter the desired name the **Name** field, and add an appropriate **Description** in the field given.
6. To configure a hook for the footer action, in the **Hook** type section, click the dropdown to **Import** or **Create New** hook.
7. In the **Page Link** section, the field provided, enter the page link from which components will be rendered within the stepper.

   Note: Pages that are used in the Page Link will not be available in the Build while adding widgets to any of the step pages.

8. To add a new step click **+ Add** and perform steps 4 to 7 to update the step properties.

   Note: The default Step 1 cannot be deleted. Only the added steps can be deleted. A delete option is provided for the added steps.

9. To change the footer action labels, in the **Footer action label change** field, click the arrow to expand the details. Enter the new labels in the fields next to **Previous**, **Next**, and **Submit**.

   Note: The footer actions are only visible in the Preview and Publish modes.

10. To add new footer action, in the **Add New Footer Action** field, click the arrow to expand the details. Enter the **Action Name** as desired. In the **Redirection link** section, choose **Internal** or **External** and enter the URL in the field provided.

    Note: Pages that are used in the Redirection Link will not be available in the Build while adding widgets to any of the step pages.

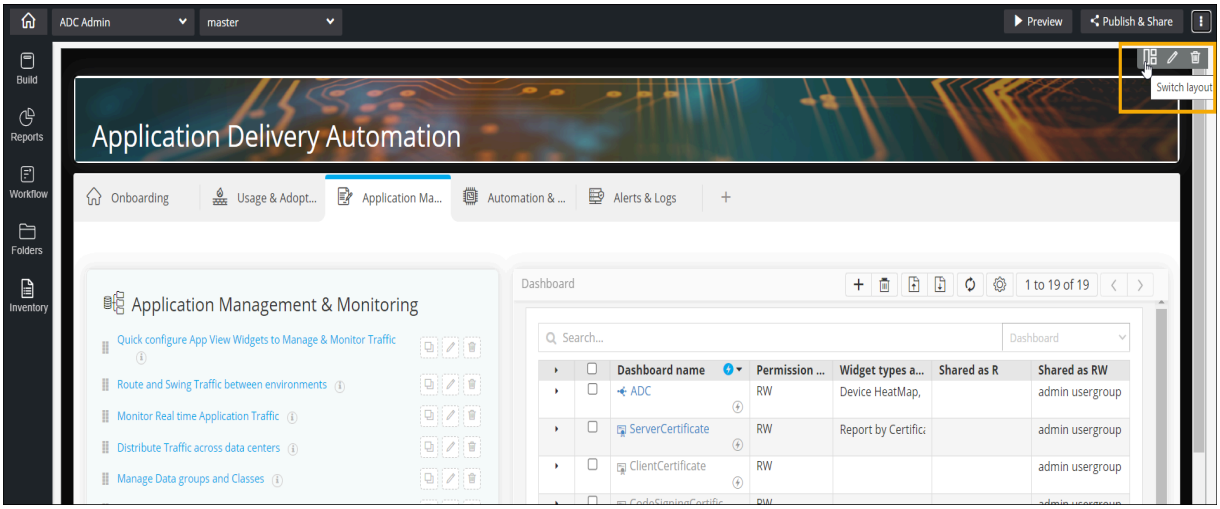
11. Click **Save**.
    The stepper properties are saved successfully.
12. To add widgets/reports to the steps pages, once you have configured the stepper properties, click **Build** in the grey left panel. Add the widgets and reports as desired.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

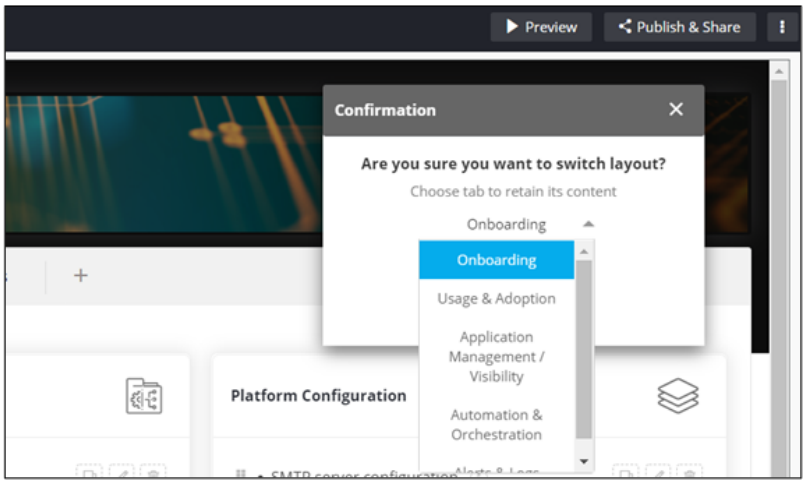
Switching from Page with tabs Layout to Single Page Layout

You can create a customized catalog with multiple tabs. There is a provision of adding up to 6 tabs in a catalog. The tabs can be arranged either horizontally or vertically.

1. In the build area, from the top right corner of the screen, click **(Switch Layout)** icon.



2. In the **Switch Layout** window that is displayed, click **Single Page**.
3. In the **Confirmation** window, from the dropdown, select the tab you want to retain on the Single page layout.

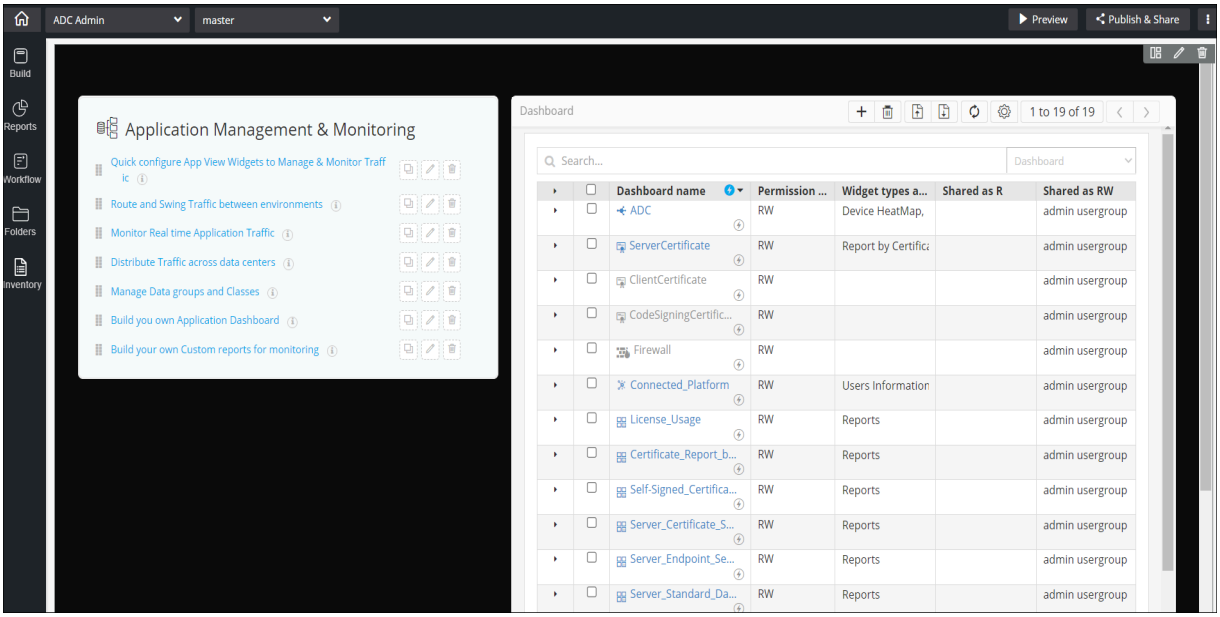


> **Warning:** Remaining tabs will be deleted and cannot be recovered even after switching back to Page with Tabs layout.

4. Once you have selected the tab to be retained, click **Yes**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The page layout is switched to Single Page displaying the retained tab.

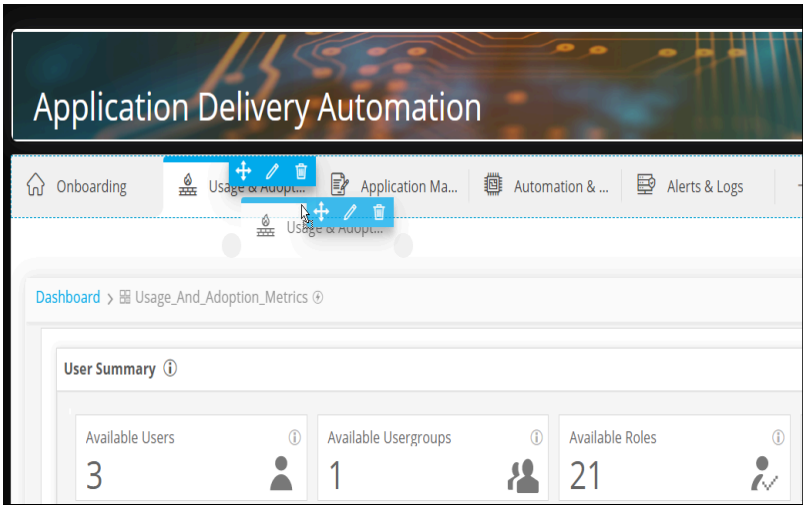


Tab reordering

When the catalog is in the Page with tabs mode, you can change the order in which the tabs are arranged.

To move the tabs:

1. Hover your mouse over a tab and click .
2. Drag and drop the tab to a new position.

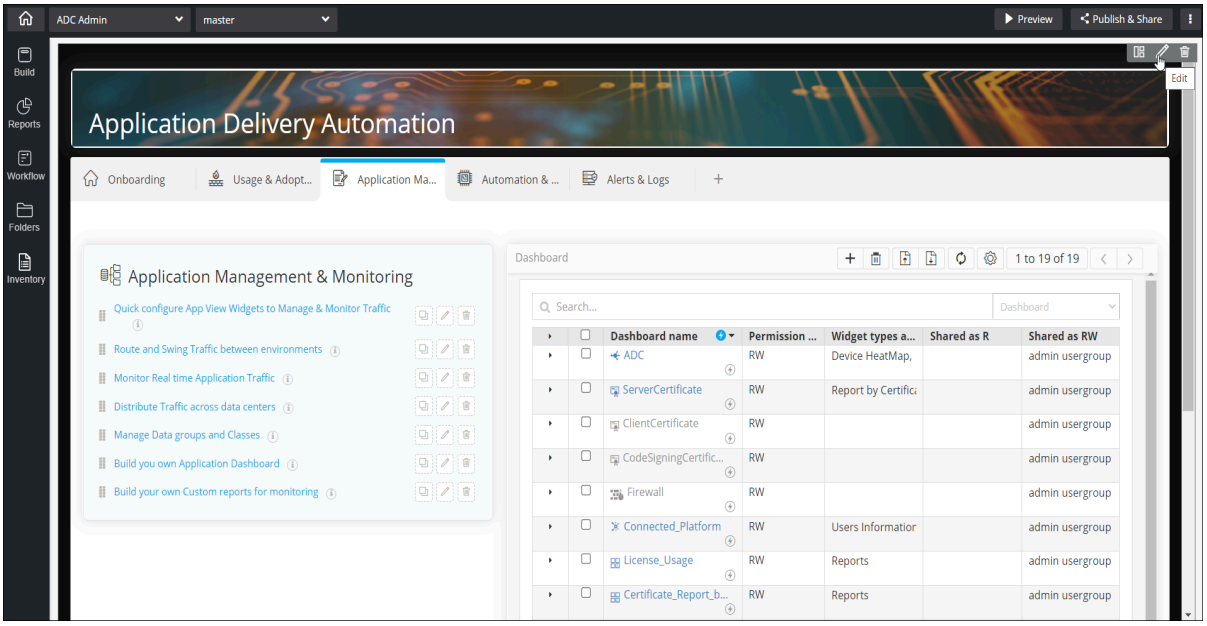


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Modifying page layout

This feature allows you to modify the layout of the entire catalog.

1. Open a page from the Design section.
2. Hover your mouse to the top right corner of the page and click **(Edit)** icon.



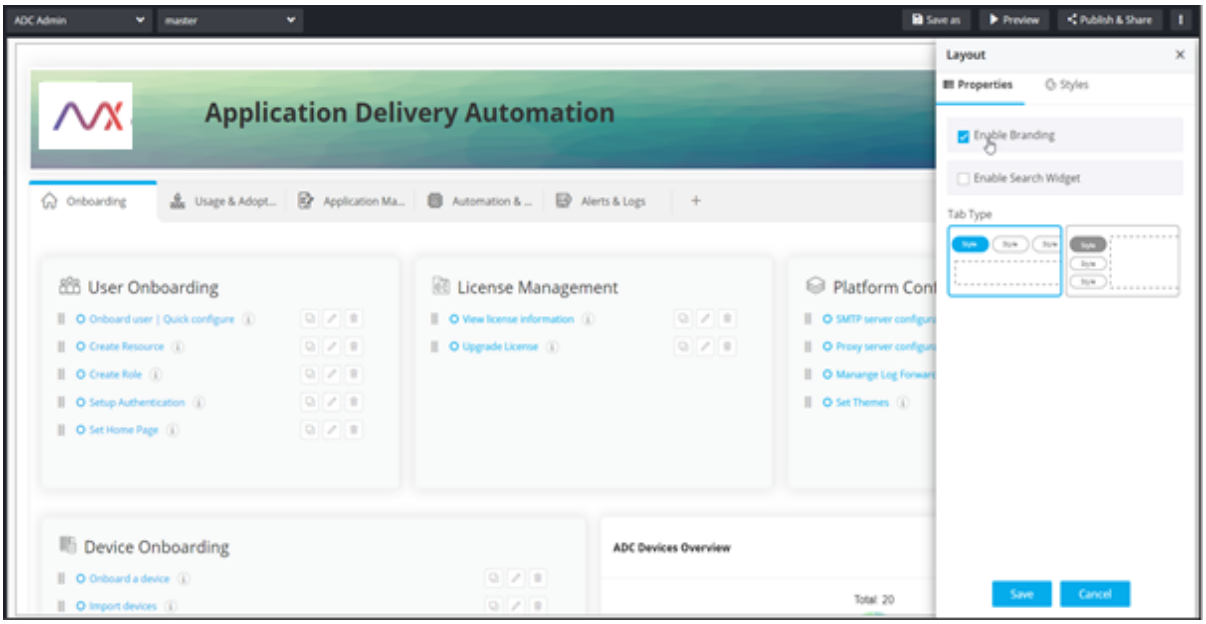
The **Layout** pop-up window is displayed with the **Properties** tab open by default.

- Modifying page layout - Properties
- Modifying page layout - Styles

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

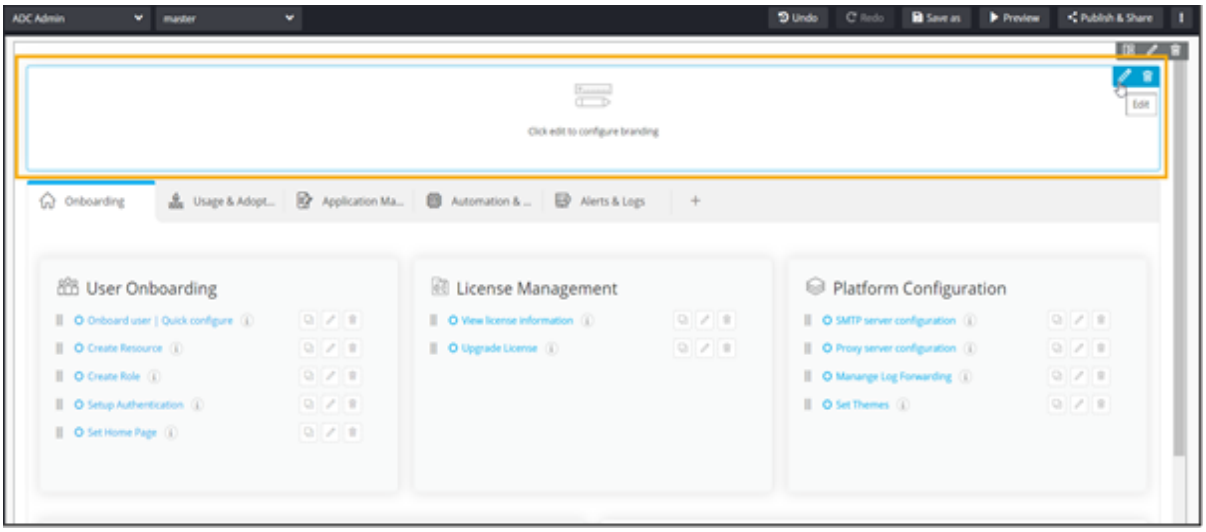
Modifying page layout - Properties

1. To enable Branding, in the **Layout** window, under **Properties**, select the **Enable Branding** checkbox.



2. Click **Save**.

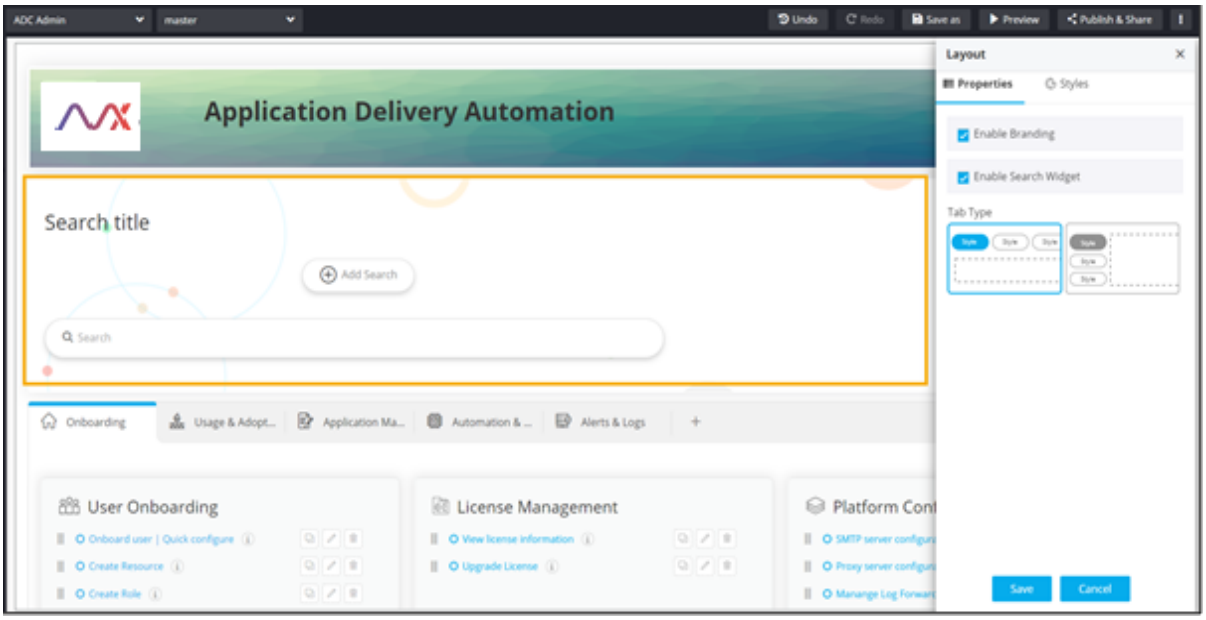
   Branding component is added to the build area.



3. Configure the Branding component.
4. To enable the search widget, in the **Layout** window, under **Properties**, select the **Enable Search Widget** checkbox.
5. Click **Save**.

   The Search widget is added to the page.

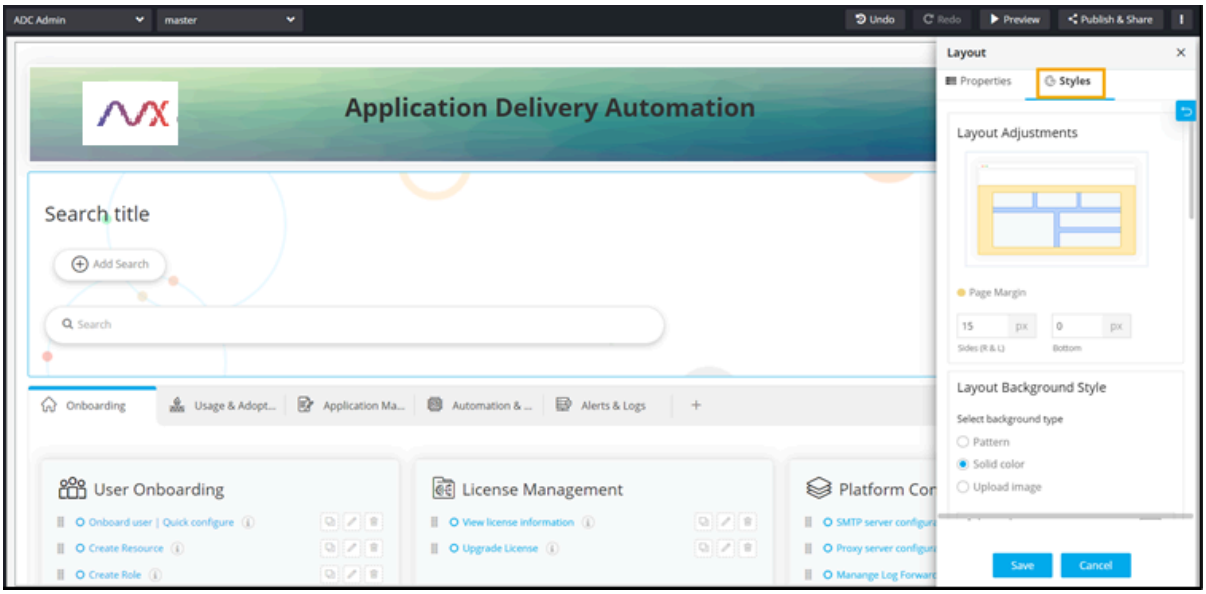
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Configure the Search widget.
7. To select the tab orientation, under **Tab Type**, select the required option.

Modifying page layout - Styles

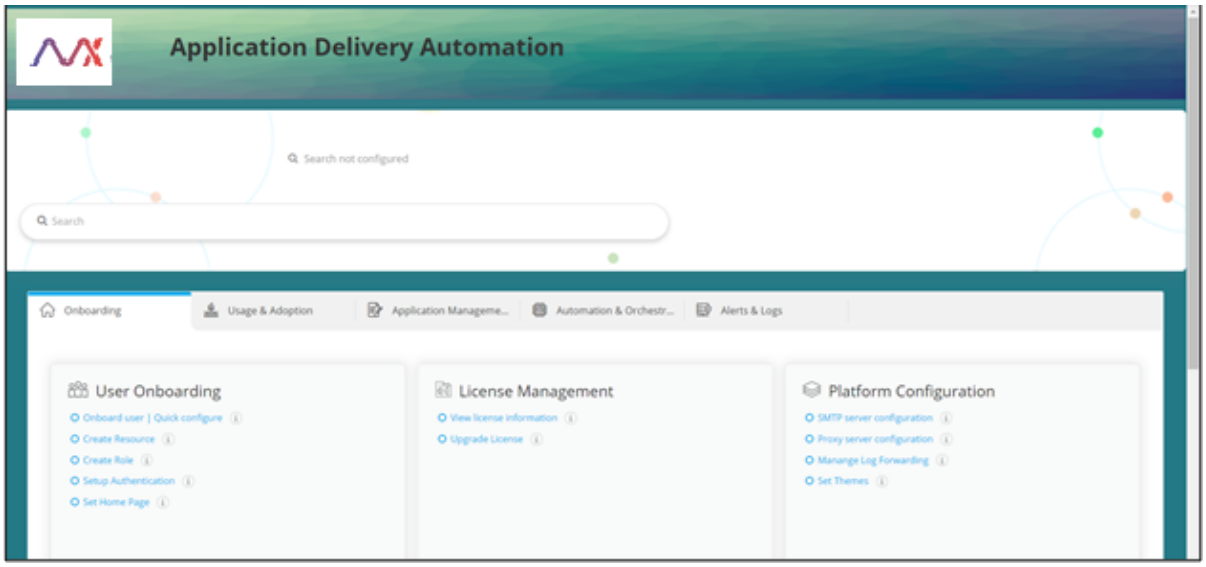
1. For more options to customize the page layout, in the **Layout** window, click **Styles**.



2. Under **Layout Adjustments**, increase/decrease the **Page Margin** at the sides and the bottom.
3. To see the changes reflected on the page, click **Save** and then click **Preview**.

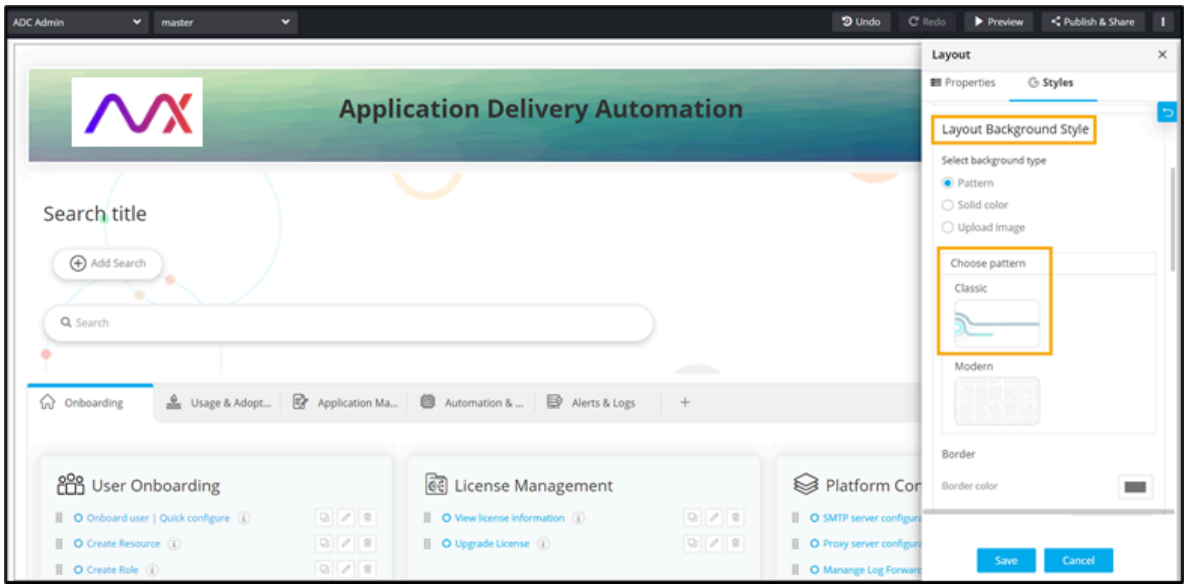
   The **Page Margin** settings have been applied to the left, right, and bottom of the page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



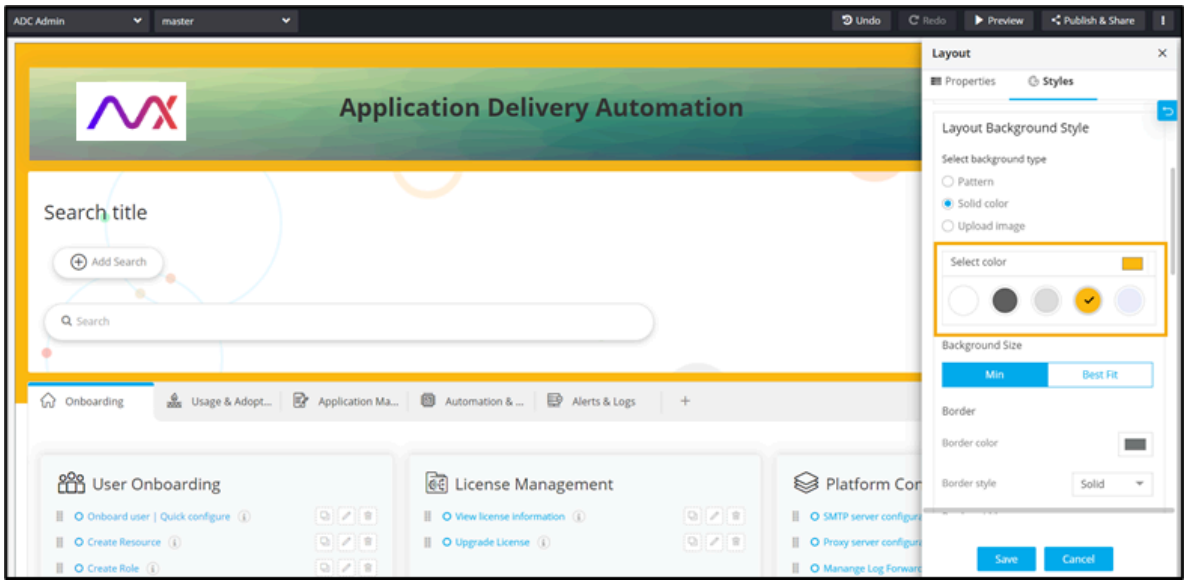
Note: Page margins are not applicable between individual widgets.

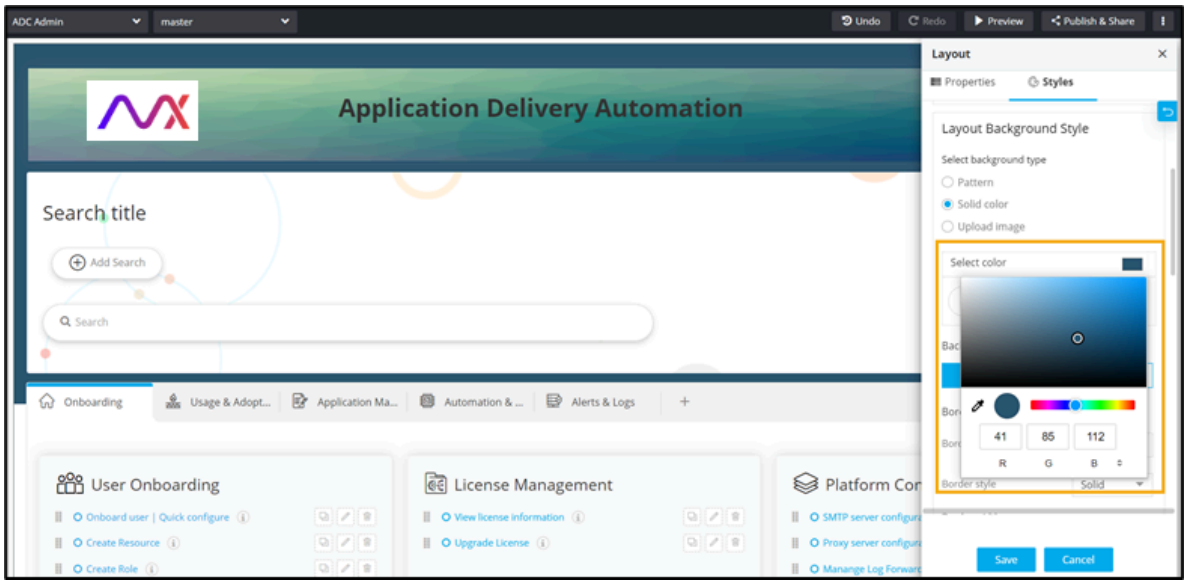
4. Under **Layout Background Style**, to change the background, **Select background type** from the available options.
   - **Pattern** - Select the **Pattern** option to choose a pattern from the options displayed.



   - **Solid color** - Select the **Solid color** option to choose a solid color either from the options displayed or from the color picker.

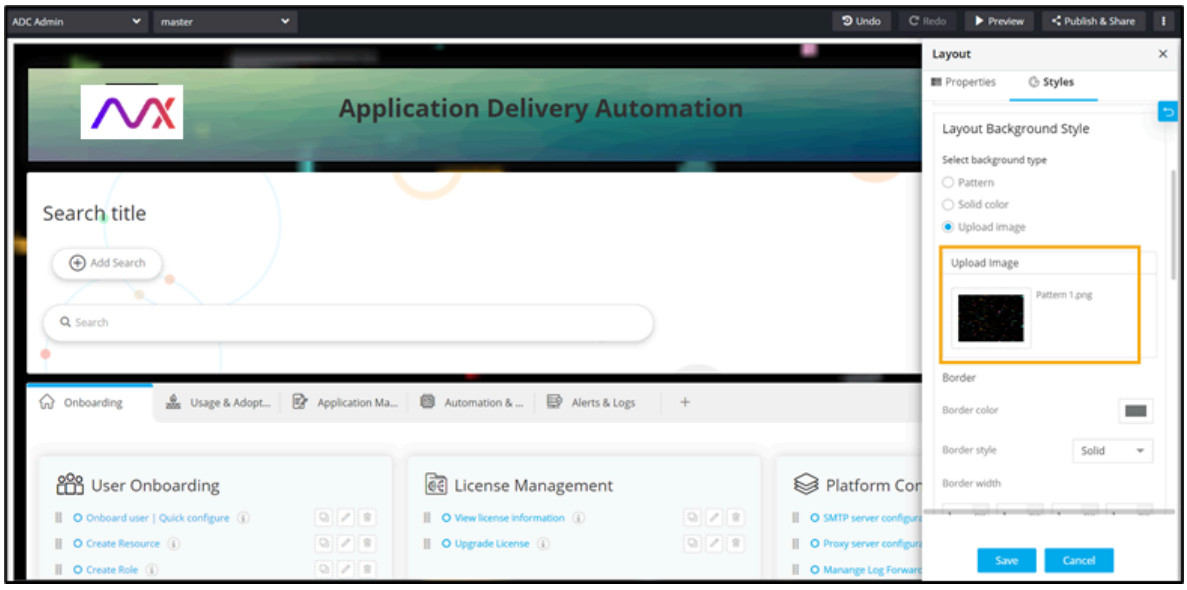
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



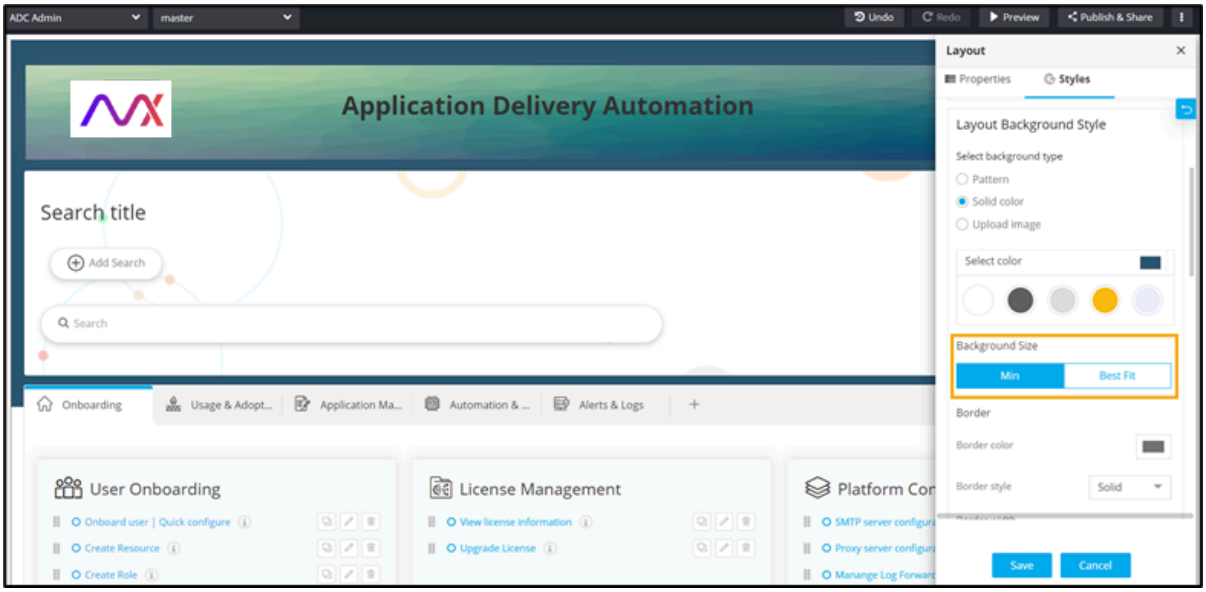


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- **Upload image** - Select the Upload image option to upload an image from your machine as the background image.



5. Under **Background Size**, to set the background color to only a part of the page, select **Min**.



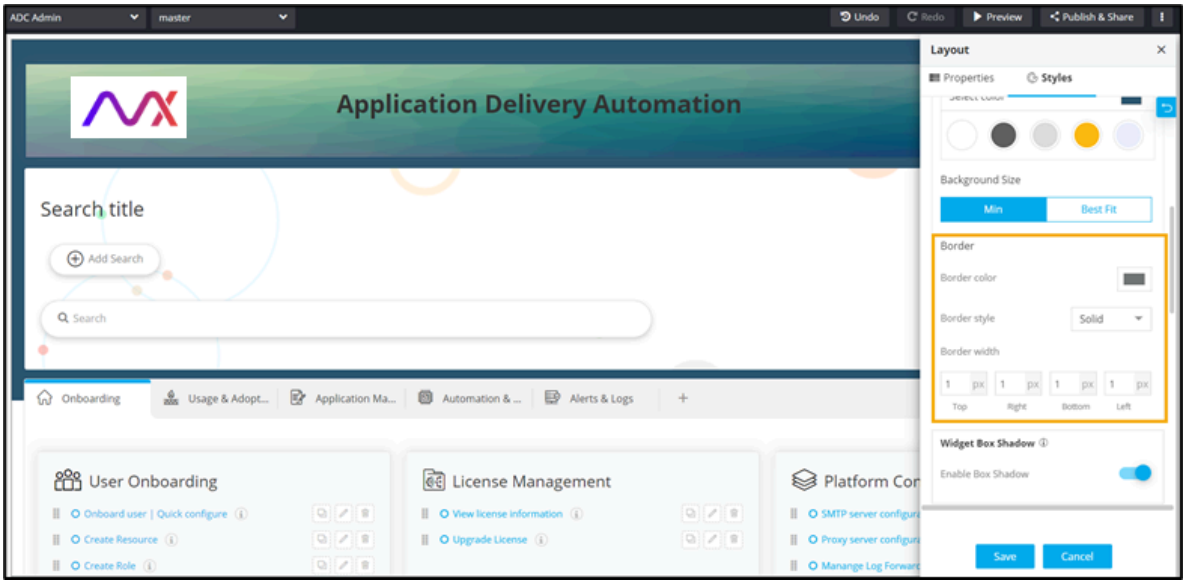
> **Note:** This feature is applicable only when the background type is selected as **Solid color**.

6. Under **Background Size**, to set the background color for the entire page, select **Best Fit**.

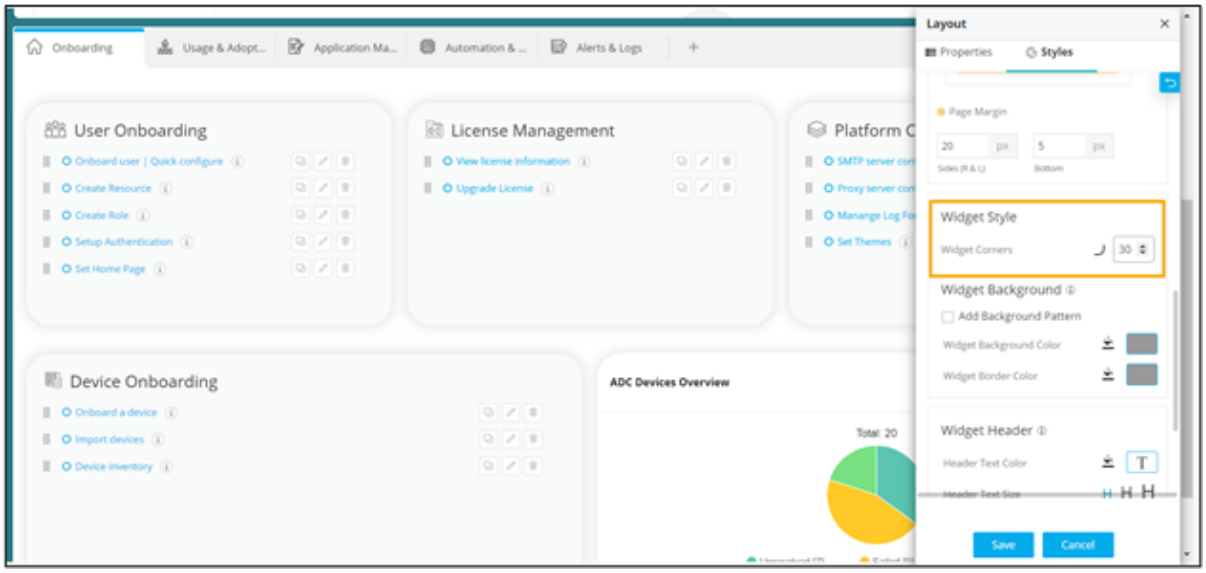
> **Note:** This feature is applicable only when the background type is selected as **Solid color**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. Under **Border** section, you can customize the following:
   - **Border color** - Select the border color from the color picker. The selected color is applied to the page border.
   - **Border style** - Select the border style from the options displayed in the dropdown.
   - **Border width** - Select the width of the border at the top, right, bottom, and left of the page.



8. Under **Widget Box Shadow**, to remove the shadow for all widgets on the page, turn off the **Enable Box Shadow** toggle.
9. Under **Widget Style**, increase/decrease the curvature of the **Widget corners**.

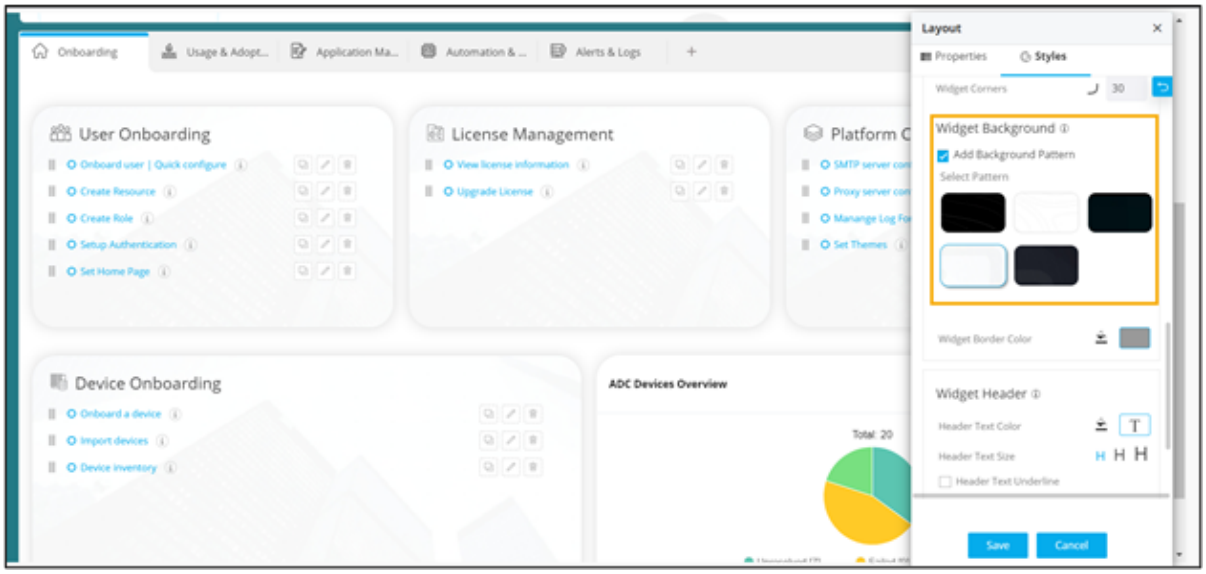


10. Under **Widget Background**, to add a background pattern, select the **Add Background Pattern** checkbox.

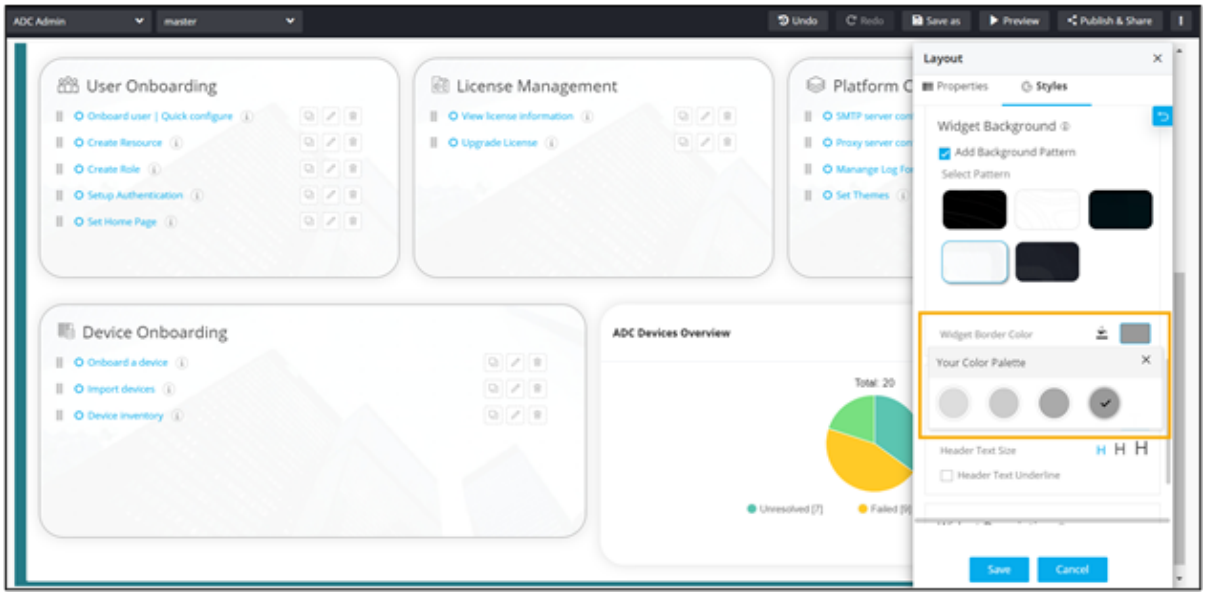
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

11. Select a pattern from the options available.

    The selected pattern is applied to all the widgets on the page.



12. Select the **Widget Border Color** from the color picker or the color palette.



13. Under **Widget Background**, select the **Widget Background Color** from the color picker or the color palette.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: The **Widget Background Color** option is displayed only when the **Add Background Pattern** checkbox is **not** selected.

14. Under the **Widget Background** section, select the **Widget Border Color** from the color picker or the color palette.
15. Under the **Widget Header** section, the following options are available:
    - **Header Text Color** - Select the **Header Text Color** from the color picker or the color palette.
    - **Header Text Size** - Select the **Header Text Size** from the available options.
    - **Header Text Underline** - Select the **Header Text Underline** checkbox to underline the Header Text.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

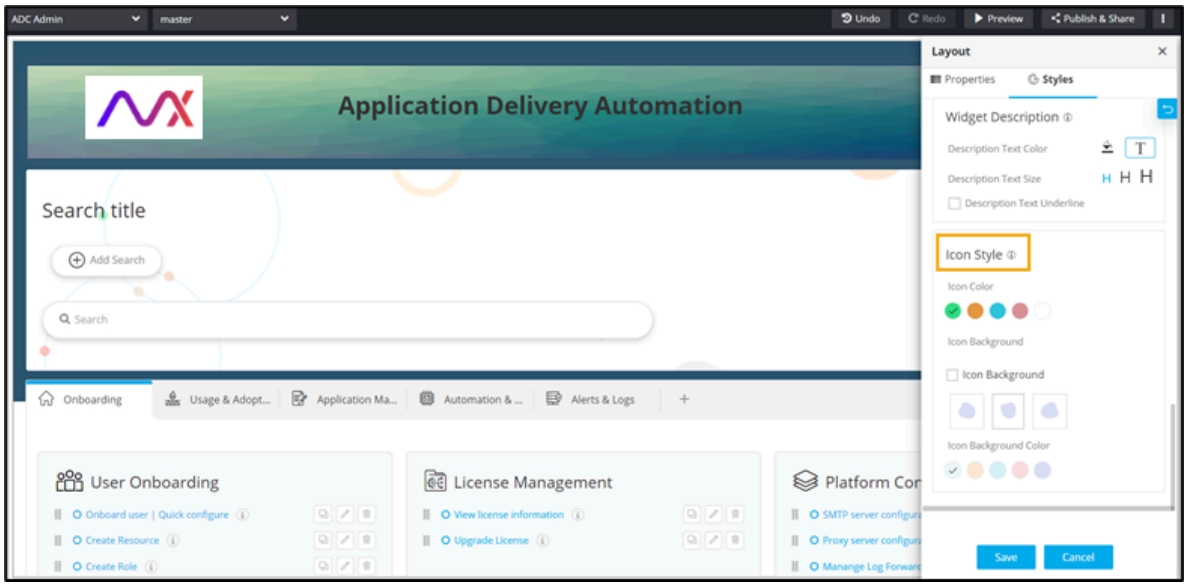
Note: Widget Header style changes are applicable to only those widgets that have a widget header.

16. Under the **Widget Description** section, the following options are available:
    - **Description Text Color** - Select the **Description Text Color** from the color picker or the color palette.
    - **Description Text Size** - Select the **Description Text Size** from the available options.
    - **Description Text Underline** - Select the **Description Text Underline** checkbox to underline the Header Text.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: Widget Header style changes are applicable to only those widgets that have a widget header.

17. Under the **Icon Style** section, choose from the following available options:
    - **Icon Color** - Select the Icon Color from the available options.
    - **Icon Background** - Select the Icon Background from the available options.
    - **Icon Background Color** - Select the Icon Background Color from the available options.

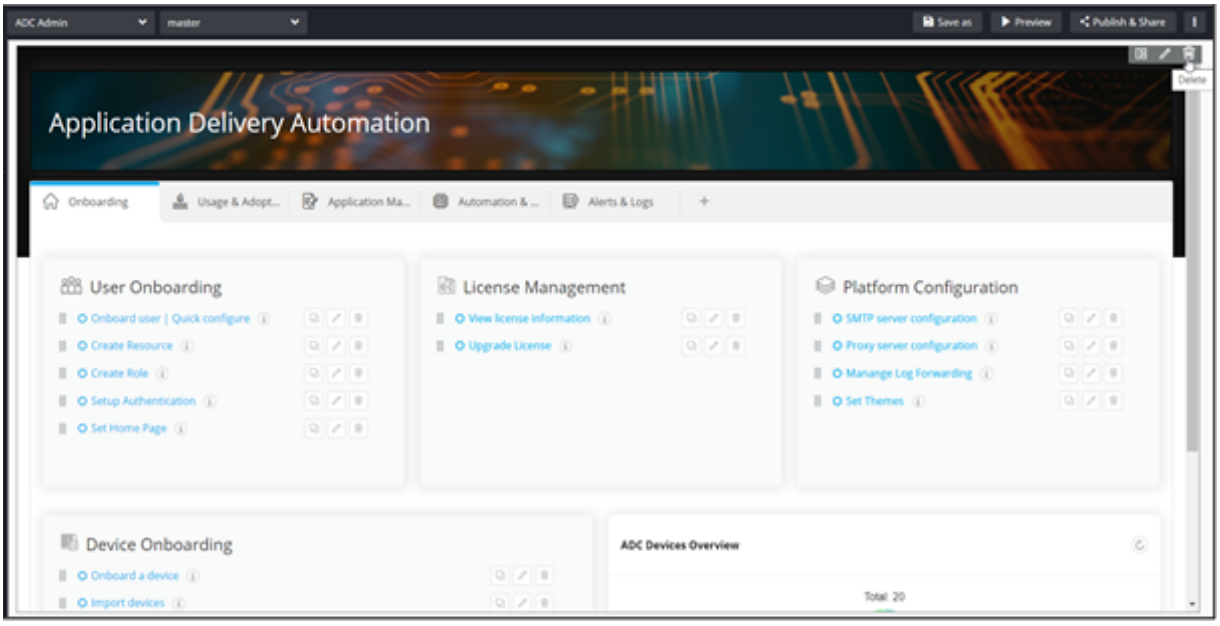


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Deleting a Catalog

You can delete the entire catalog in one go.

1. Form the Design section, open the page that is to be deleted.
2. Hover your mouse to the top right corner of the page and click **(Delete)** icon.



3. In the **Confirmation** pop-up window, click **Yes**.
   The catalog is deleted.

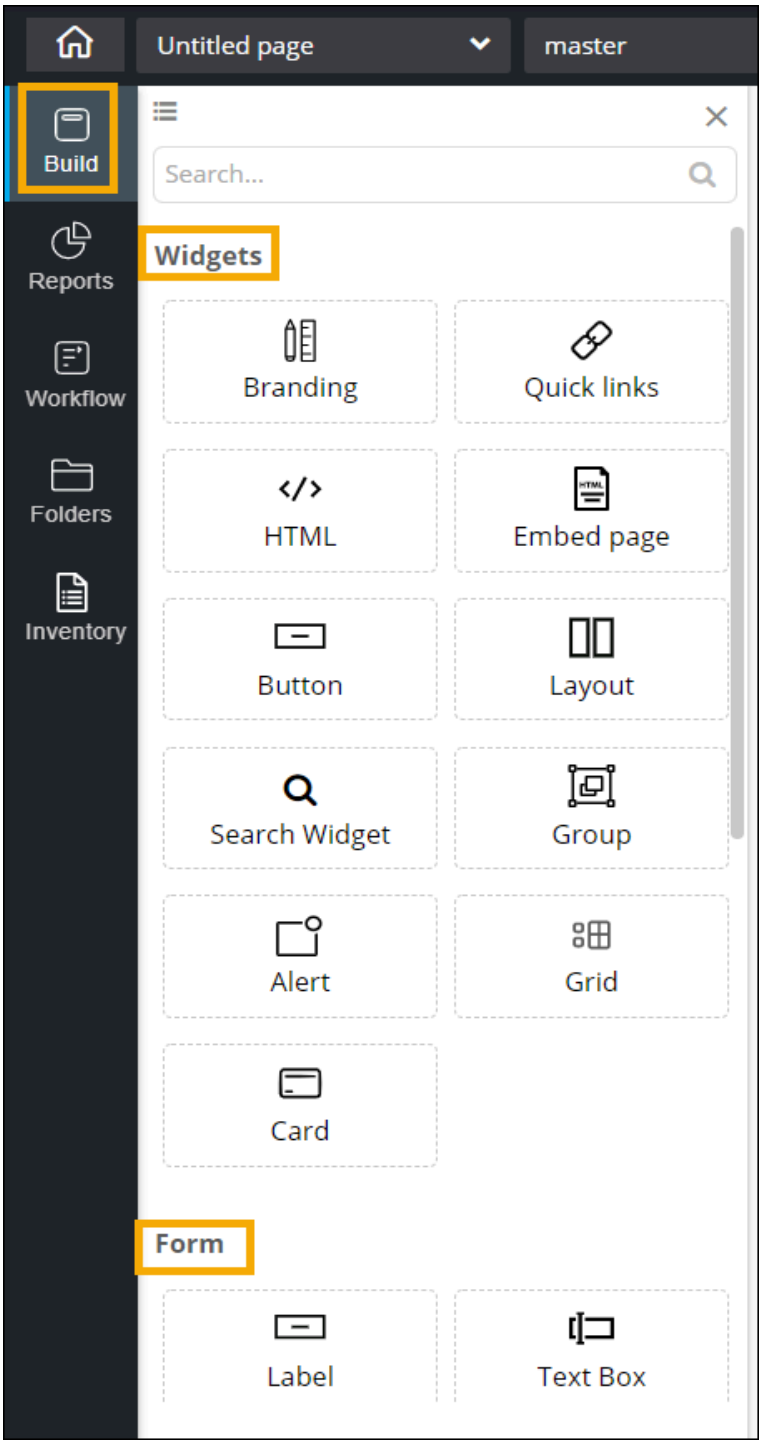
> **Important:** This action cannot be undone.

Build

The Service Catalog allows you to build your custom page by using the Build component. You can simply drag and drop components from this section to design a customized page.

1. Design a new page.
2. From the component panel on the left, click **Build**.
   A list of Page elements is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Options available and Description

Group	Option	Description
		Allows you to toggle between a list view and grid view of the components.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Group	Option	Description
	Search bar	Allows you to search for a particular component.
Widgets		
	Branding	Allows you to configure branding for your page.
	Quick links	Allows you to add a quick links widget to your page.
	HTML	Allows you to create your own HTML page within the Page.
	Embed page	Allows you to Embed a page into the Page.
	Button	Allows you to add buttons for redirecting to internal/external web pages from the catalog.
	Layout	Allows you to arrange widgets within a layout.
	Search widget	Allows you to add a search widget to the page.
	Group	Allows you to create groups for clubbing similar components/widgets together in the Page.
	Alert	Allows you to add and view Visual Workflow Alerts in the Page.
	Grid	Allows you to add a Grid component to your Page.
	Card	Allows you to add a card component with option to add multiple links, buttons and content.
Form		
	Label	Allows you to add a Label to the form in your Page.
	Text Box	Allows you to add a Text Area to the form in your Page.
	Password	Allows you to add a Password field to the form in your Page.
	Date Time Picker	Allows you to add a Date Time Picker field to the form in your Page.
	Text Area	Allows you to add a Multi-select Box to the form in your Page.
	Radio Button	Allows you to add a Radio button to the form in your Page.
	Checkbox	Allows you to add a Checkbox to the form in your Page.
	Select Box	Allows you to add a Select Box to the form in your Page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Group	Option	Description
	Multi-select Box	Allows you to add a Multi-select Box to the form in your Page.
	Form Button	Allows you to add a Form Button to the form in your Page.
	Download	Allows you to add a Download field/link to the form in your Page.
	File Upload	Allows you to add a File Upload field to the form in your Page.
	Script Editor	Allows you to add a Script Editor to the form in your Page.
	Tabular	Allows you to add a tabular feature to the form in your Page.
	Form Group	Allows you to define a Form Group to your form in the Page.
	Rich Text Editor	Allows you to add a text editor to the form in your Page.

- Widgets
- Form

Widgets

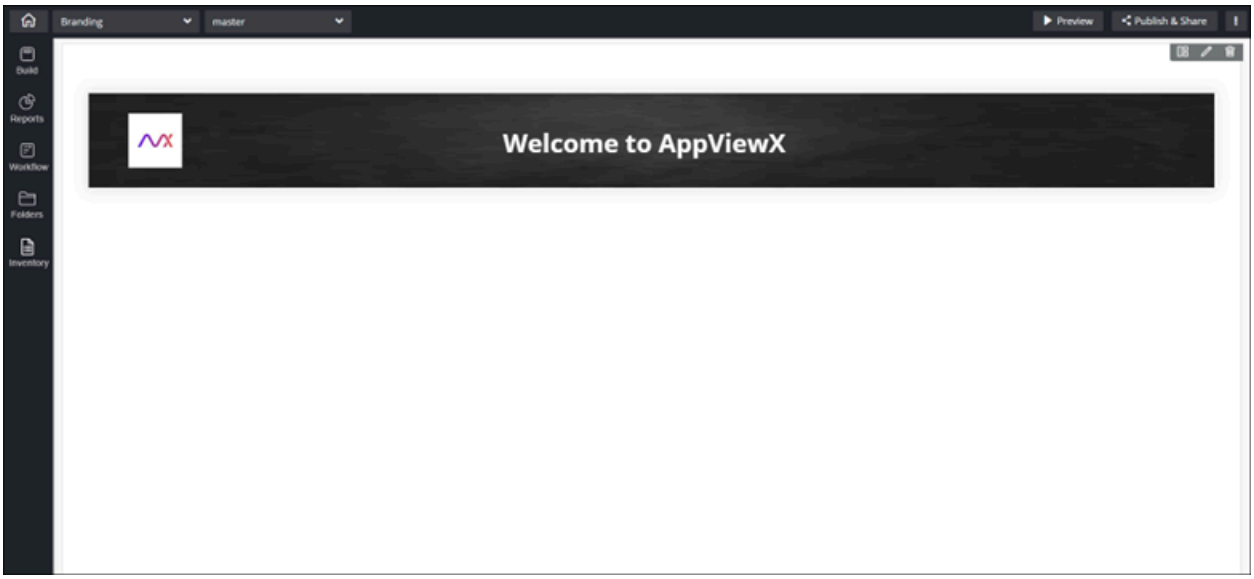
You can add different elements to your page in the form of widgets, configure them, and customize them as per your requirement.

- Branding
- Quick Links
- HTML
- Embed page
- Button (Action Widget)
- Layout
- Search Widget
- Group
- Alert
- Grid
- Card

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

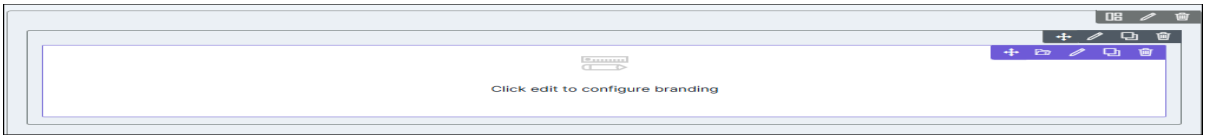
Branding

Branding or Header component allows user to create a banner showcasing the title or main objective for which the page is created.



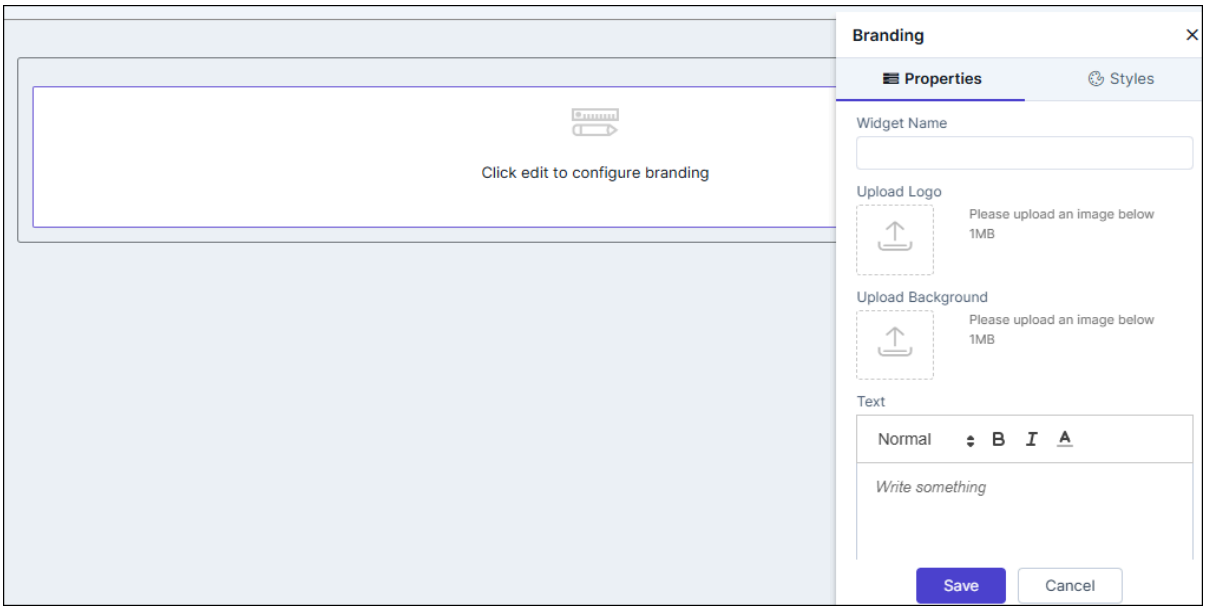
Configuring Branding - Properties

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Branding** widget to any blue highlighted space on the canvas.
2. To configure the Branding widget, hover your mouse over the widget and click **(Edit)** icon.



   The **Branding** pop-up window opens with the **Properties** tab open by default.
3. In the **Branding** window, under **Properties**, enter or select the required fields.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



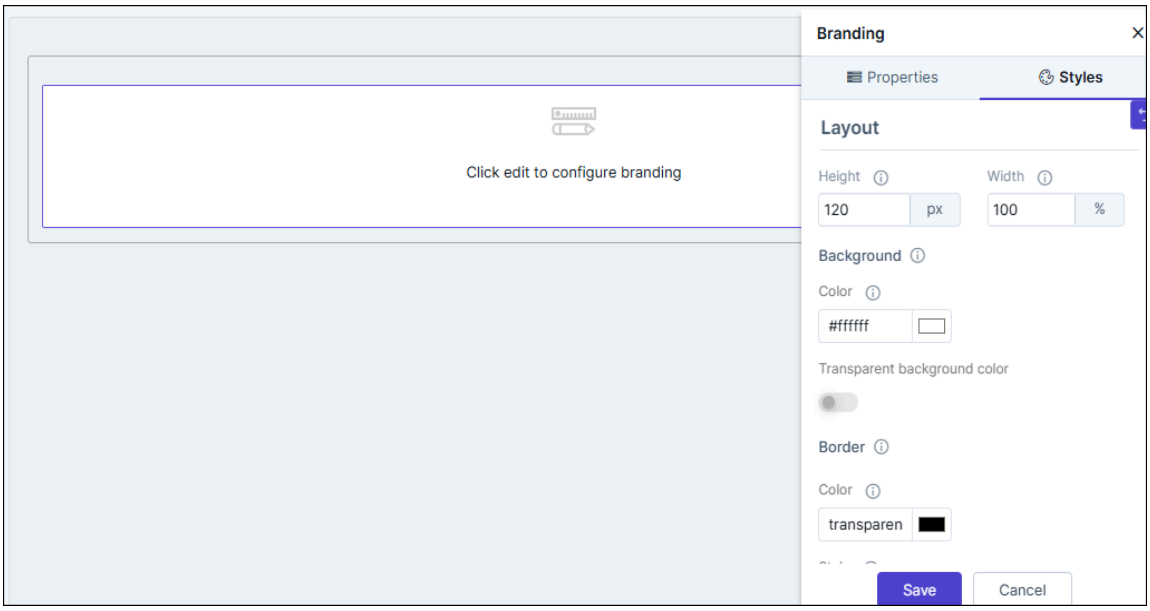
Field descriptions for Properties

Field	Description
Widget Name	Enter a suitable widget name. For example, AppViewX.
Upload Logo	The branding or header widget allows users to upload their logo or any custom image that represents their personal brand.
Upload Background	Users can set a background image for the branding or header widget, adding a unique visual element to the page.
Text	In addition to the logo or image, users can add custom text to display in the header section, further conveying their brand message or tagline.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring Branding - Styles

1. For more options to customize the widget, in the **Branding** window, click **Styles**.



Field descriptions for Styles

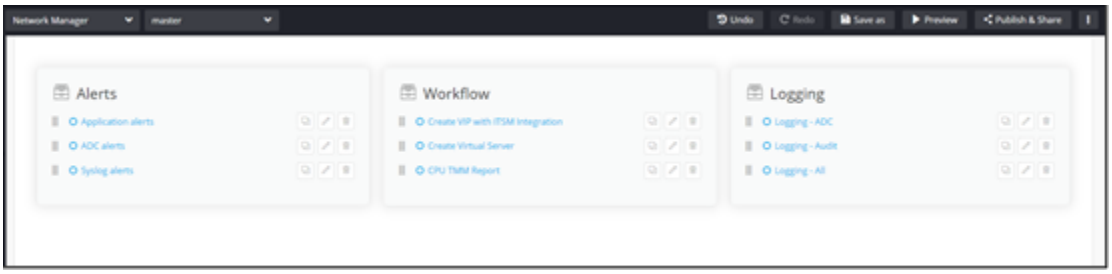
Option	Description
Sizing	Allows you to increase/decrease the height and width of the widget.
Background	The following options are available here: • **Background color**: Allows you to select a background color for the branding widget. • **Set transparent background color**: To set the background color as transparent, turn on the toggle.
Border	The following options are available here: • **Border color**: Allows you to select the border color. • **Border style**: Allows you to select the border style from the options available in the dropdown, e.g. dotted, dashed, inset etc. • **Border width**: Allows you to adjust the width of the border.

2. Click **Save**.
3. Drag and drop the background image and text to move it to any position within the widget and resize the company logo.
4. To restore settings to default, click **(Restore)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

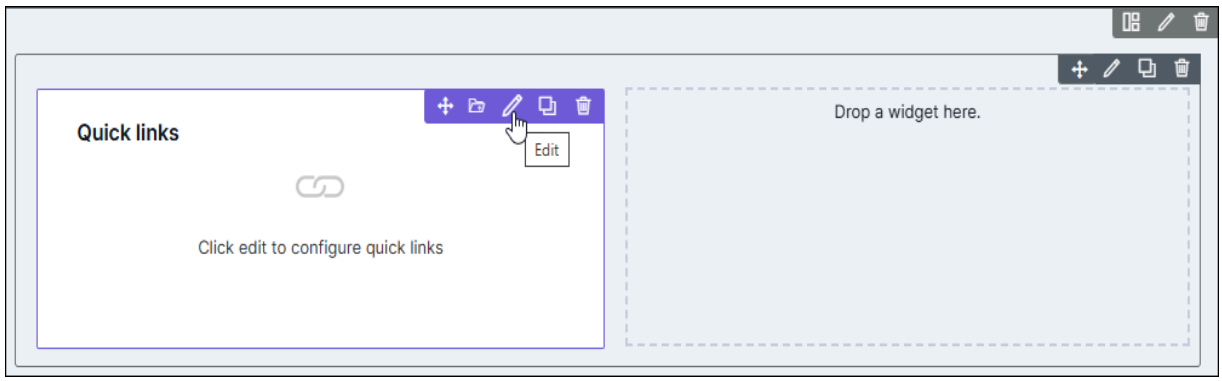
Quick Links

The Quick Link widget provides a convenient way for users to add easy-to-use hyperlinks to their pages. You can add Quick links to your page for redirecting end users to specific features within AppViewX (ADC, Certificates, Reports etc.) and outside AppViewX (JIRA etc). Clicking on a Quick link within the widget will redirect you to a new tab.



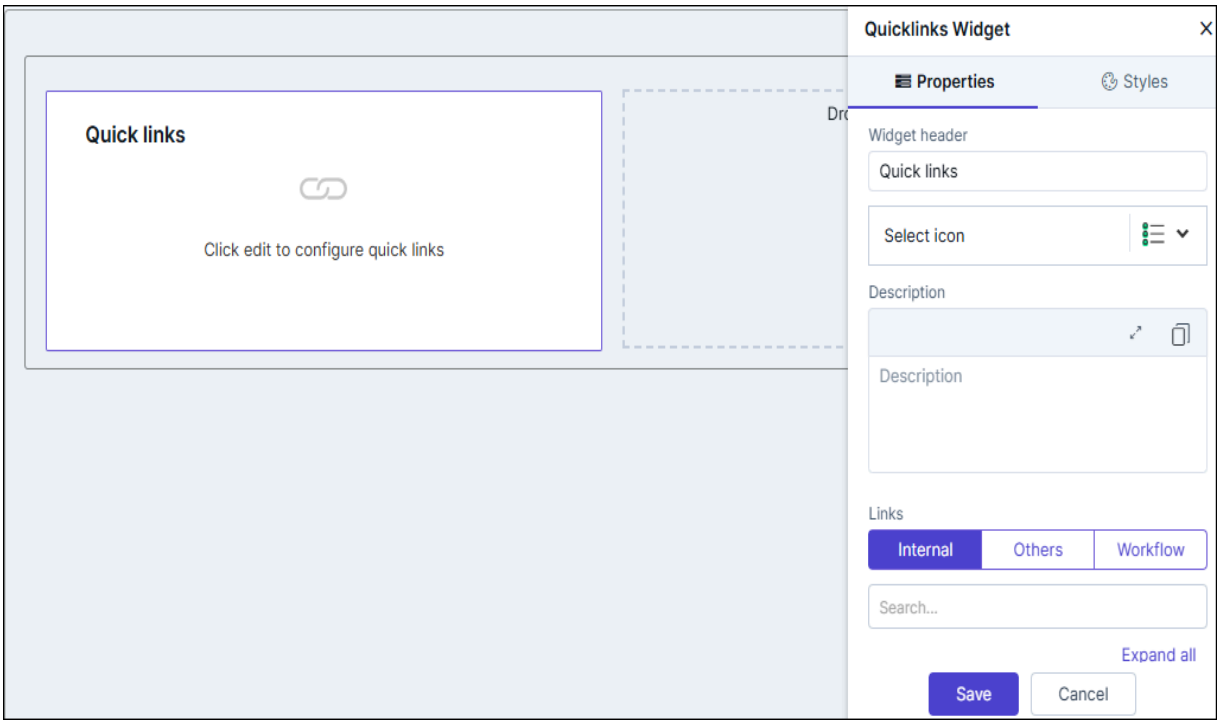
Configuring Quick Links - Properties

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Quick links** widget to any blue highlighted space on the page.
2. To configure the Quick links component, hover your mouse over the widget and click **(Edit)** icon.



   The **Quicklinks Widget** pop-up window opens with the **Properties** tab open by default.
3. In the **Quicklinks Widget** window, under **Properties**, enter or select the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Properties

Field	Description
Widget Header	Provide a name for the Widget Header. For example: **Quick Links**.
Select icon	Select an icon for the Quick Links widget from the dropdown.
Description	Users can add a brief description for the widget to provide context or additional information.

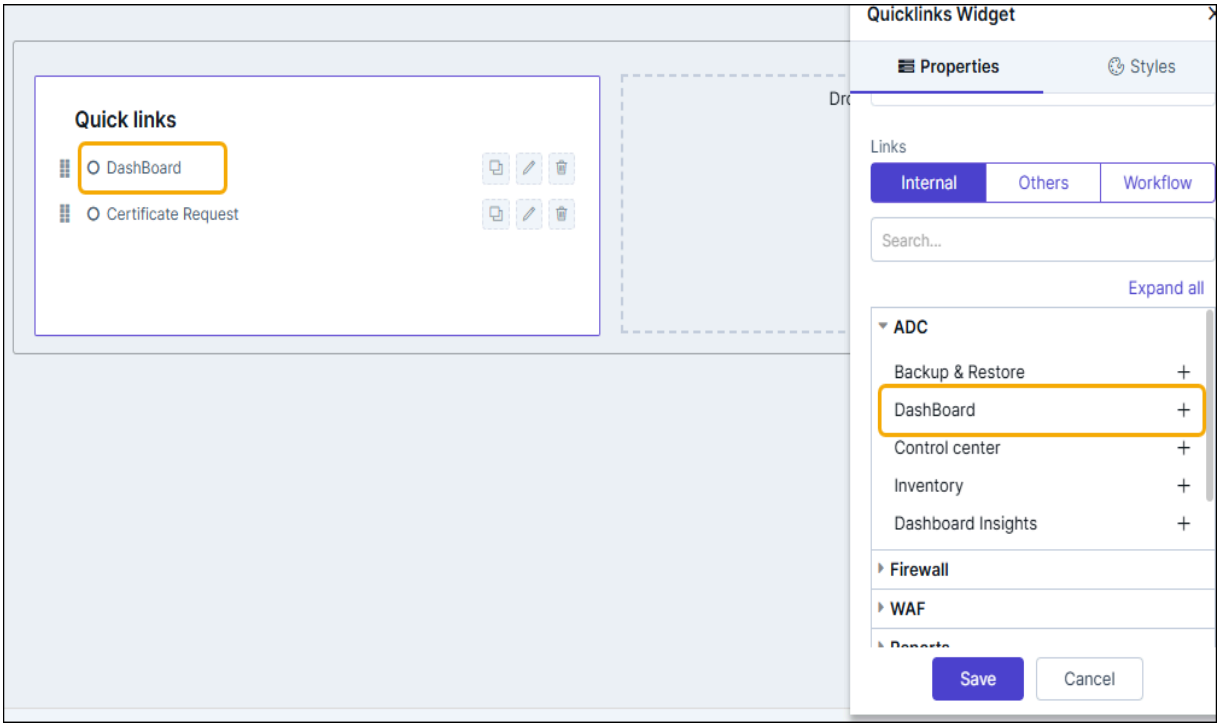
Add Links	You can add three types of links here:
	• **Internal**: Users can choose from pre-packaged hyperlinks that are already available within the application.
	• **Others**: In this option, users can add two types of links: Internal and External.
	  • **Internal:** Links that navigate to specific locations within the application.
	  • **External:** Links that redirect to external web pages or resources outside the application.
	• **Workflow**: Users can create links to specific service catalogs or workflows, allowing them to seamlessly access relevant workflows directly from the page.

4. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Adding an Internal Quick Link

1. To link an internal AppviewX page to the widget, under **Add links**, click **Internal**.
2. To select a link to be added to the widget, click **(Add)** icon.

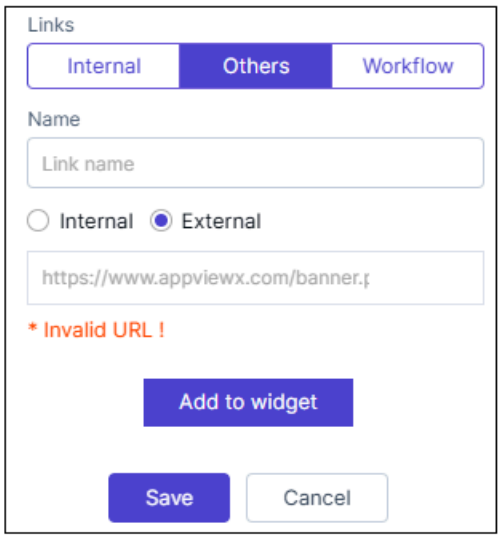


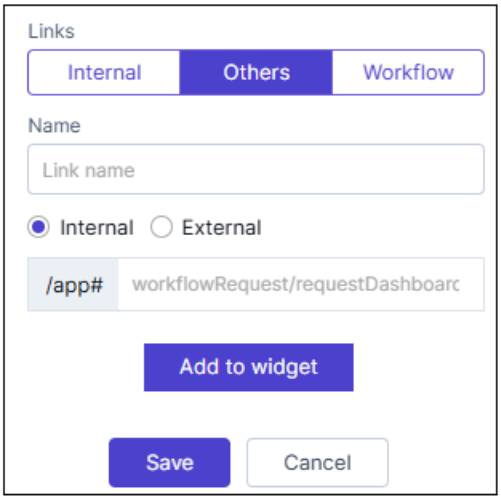
3. To add selected links to the widget, click **Save**.
   Clicking on the quick link will redirect to the selected page in a new tab.

Adding Others Quick Links

1. To embed an Internal link or a cross origin enabled page URL into the Quick link widget, under **Links**, click **Others**.
2. Enter the **Name** and **URL** for the link.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



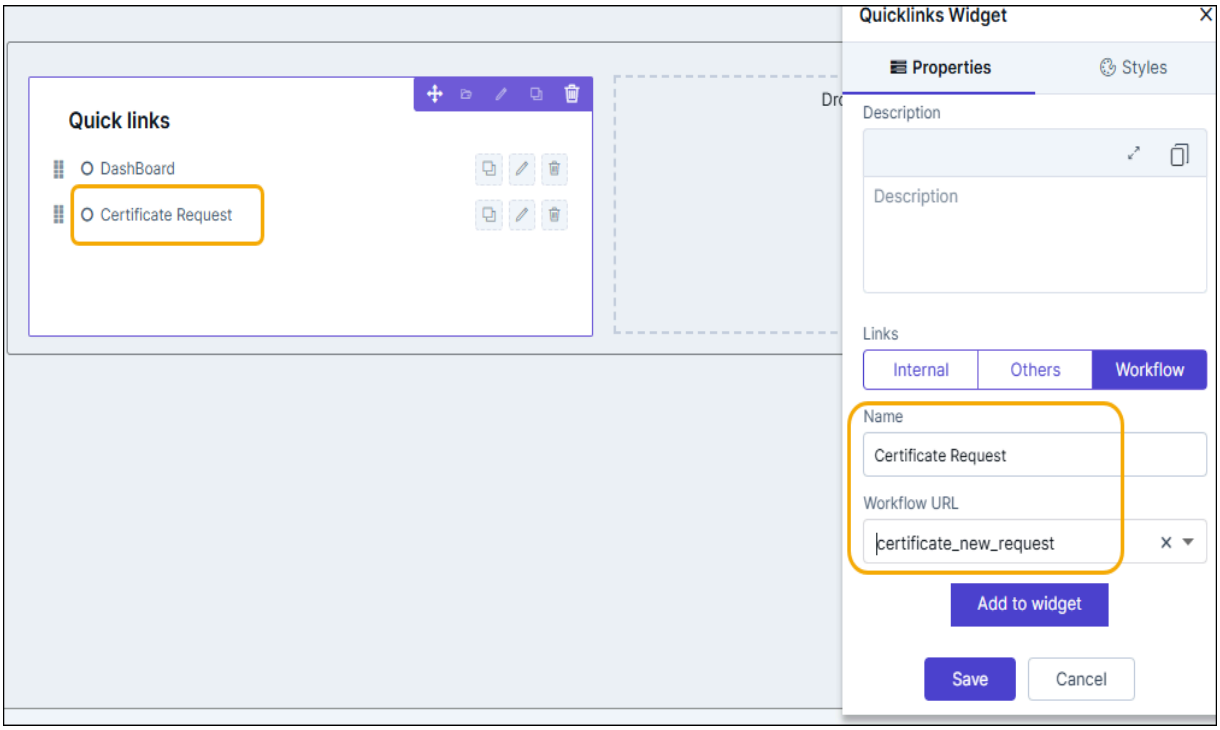


3. Add the internal or external links as follows:
   a. Select the **Internal** radio button, and append the URL to **/app#** field. (these are the AppViewX configured URLs/pages)
   b. Select the **External** radio button and enter the complete URL in the field. example: *www.google.com*.
4. To add this external link to your widget, click **Add to Widget**.
   Clicking on the quick link will redirect to the configured external page.

Adding a Workflow Quick Link

1. To add a link to redirect to the Workflow Request execution page for the selected workflow, under **Add links**, click **Workflow**.
2. Enter the link **Name** and select the **Workflow URL** from the dropdown.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

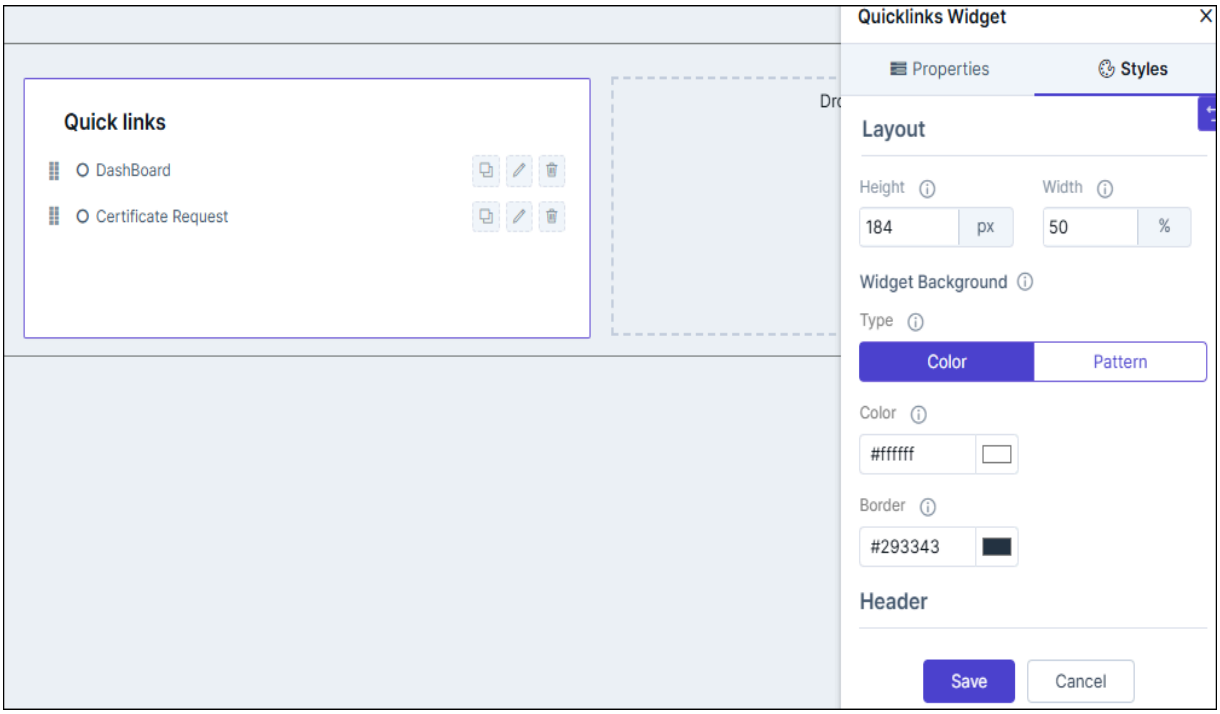


3. To add this workflow URL to the Quick Links widget, click **Add to Widget.**
   Clicking on the quick link will redirect to the workflow request execution page of the selected workflow.
4. Once you have added the required workflow URLs to the widget, click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring Quicklinks - Styles

1. For more options to customize the widget, in the **Quicklinks Widget** window, click **Styles**.



Field descriptions for Styles

Field	Description
Sizing	Allows you to increase/decrease the height and width of the widget.
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, unselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

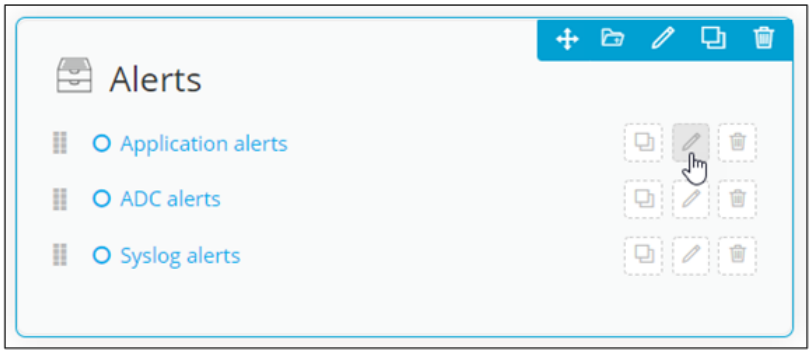
Field	Description
Widget Description	The following options are available here: • **Description Text Color**: Allows you to change the color of the label text. • **Description Text Size**: Allows you to select the size of the label text. • **Description Text Underline**: To underline the description text, select the **Description Text Underline** checkbox.
Icon Style	The following options are available here: • **Icon Color** - Select the Icon Color from the available options. • **Icon Background** - Select the Icon Background from the available options. • **Icon Background Color** - Select the Icon Background Color from the available options.

2. To save your settings, click **Save**.
3. To restore the settings to default, click **(Restore)** icon.

Editing and Deleting Quick Links

Individual quick links can be edited to change the name displayed on the widget, add help text to quick links, and also delete quick links from the widget.

1. To edit an internal or external Quick link click **(Edit)** icon.



2. Enter the **Alias name** in the quick link pop-up window.
3. To edit the Workflow quick link, click **(Edit)** icon.
4. In the workflow quick link pop-up window, edit the workflow name and select a different workflow URL from the dropdown list.
5. To display the help text when you hover your mouse over a quick link, select the **Help info** checkbox and add the text to be displayed.
6. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To delete a quick link, click **(Delete)** icon and click **Ok** in the **Confirmation** window.
8. To rearrange the order in which the quick links are displayed in the widget, hold and drag the quick link where required.
9. To make a copy of a quick link, click **(Clone)** icon.

HTML

The HTML widget is a powerful tool that allows users to create their own custom components by writing HTML, CSS, and JavaScript code. With this widget, users can build unique components with their desired data and style properties. The HTML widget offers the same layout properties as other components, providing flexibility and consistency in page design. You can design customized HTML pages based on specific user needs.

- Create customized HTML page using HTML5 syntax
- Add third party web plugins
- Add own style components and interactive functionalities using CSS and JS
- Use Hooks to design custom HTML pages.

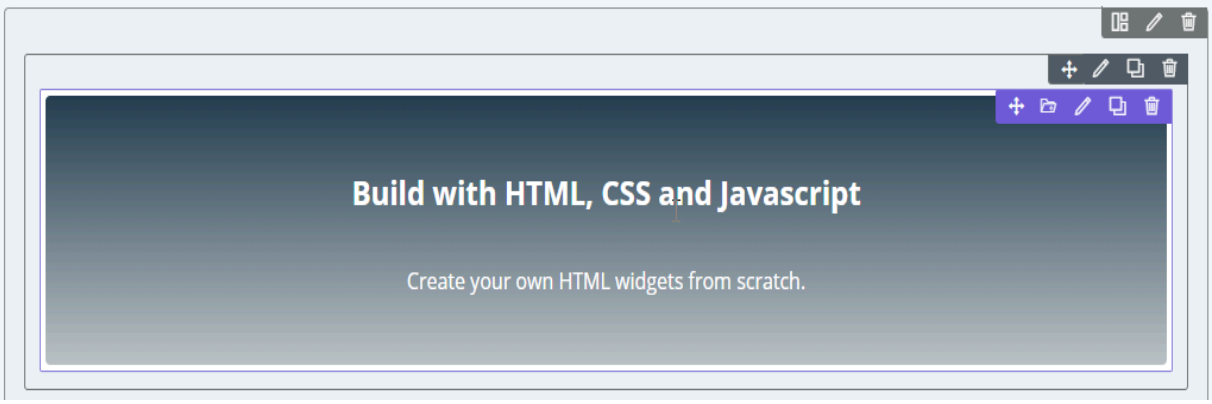
Additional Features

- **jQuery and Bootstrap Support:** The HTML component supports jQuery and Bootstrap libraries, enabling users to leverage the functionality and styling options provided by these popular frameworks. This integration allows users to enhance the interactivity and responsiveness of their custom components.
- **Sample Code:** The HTML widget includes a set of sample code snippets showcasing how to use jQuery and Bootstrap features within the component. These samples serve as a reference and starting point for users to explore and experiment with different functionalities.
- **Hooks Integration:** Users can make use of existing hooks from the Hook Inventory or create new hooks to enhance the functionality of the HTML component. Hooks allow users to encapsulate and reuse code logic, promoting modularity and efficiency in page building.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring HTML Widget - Properties

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **HTML** widget to any blue highlighted space on the page.
2. To configure/edit the HTML widget, hover your mouse over the widget and click **(Edit)** icon.



The **Html widget** pop-up window opens with the **Properties** tab open by default. You can now create HTML pages using:

- Third-party Plugins
- Prebuilt Samples
- Hooks

Creating Custom HTML Pages using Plugins

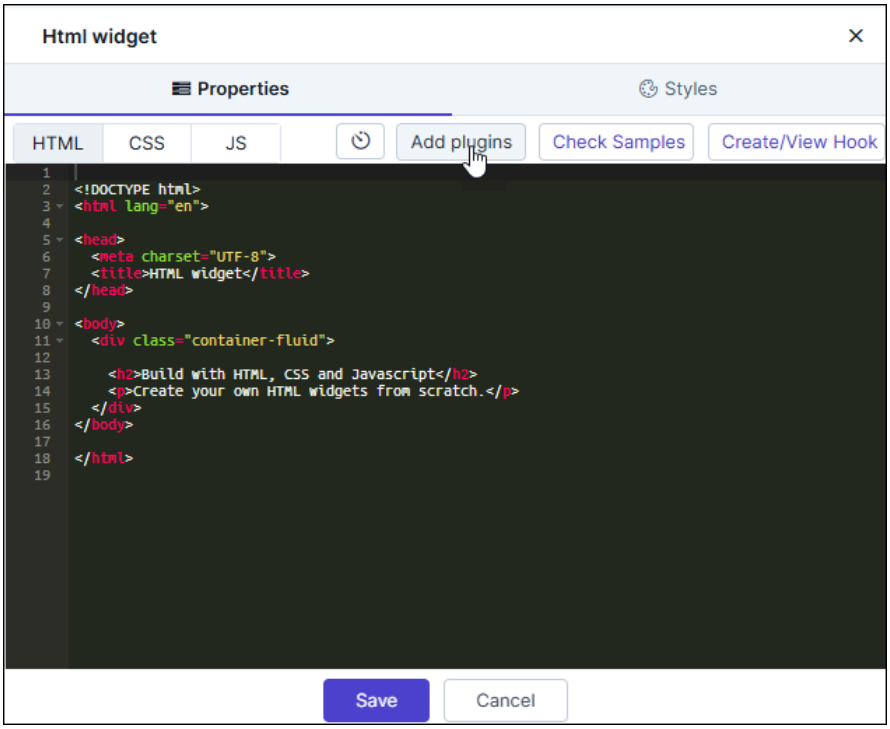
Add third party web plugins to the HTML widget; the following plugins are supported natively:

- jQuery
- Bootstrap.

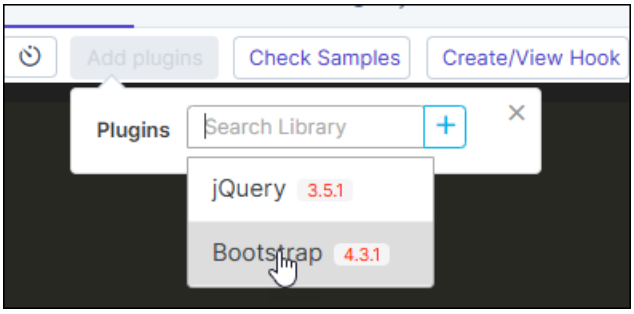
To add plugins to your HTML page:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

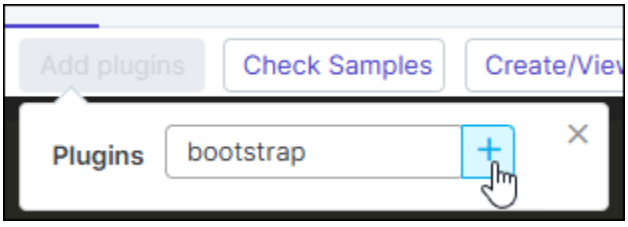
1. In the **HTML widget** window, under **Properties**, click **Add plugins**.



2. Select the library from the available options. For example, **Bootstrap**.

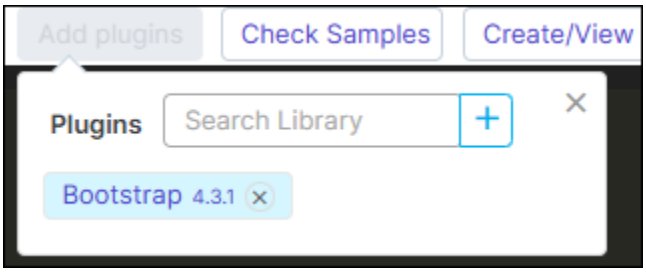


3. To add the selected plugin, click **(Add)** icon.



4. To see the available design templates, click **Bootstrap**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The Bootstrap library opens in a new tab. You can customize any of their available templates for your HTML page.

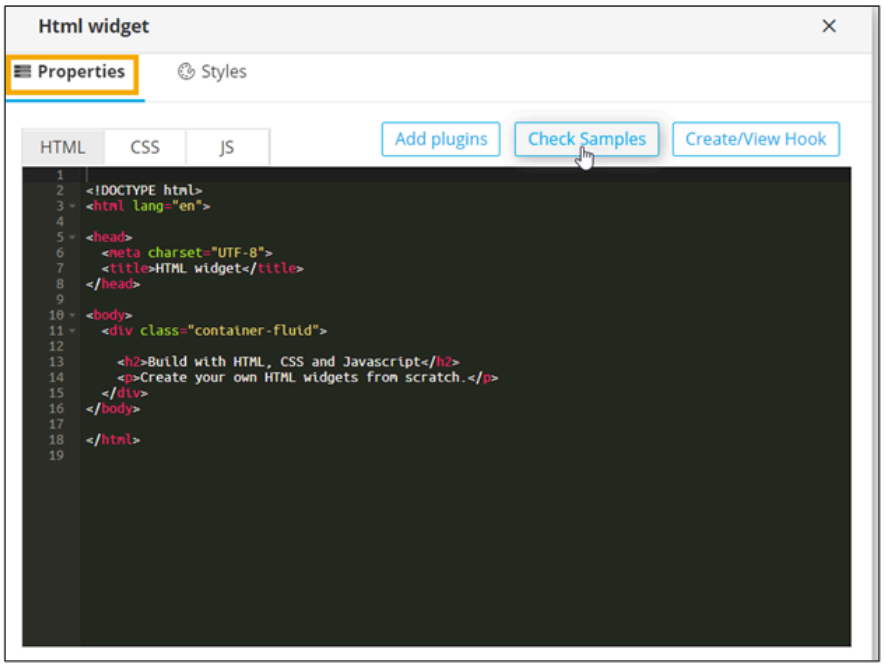
5. To save your selection, click **Save**.

Creating Custom HTML Pages using Sample HTML Script

Create an HTML widget in your Page using prebuilt HTML scripts provided within the Page Builder.

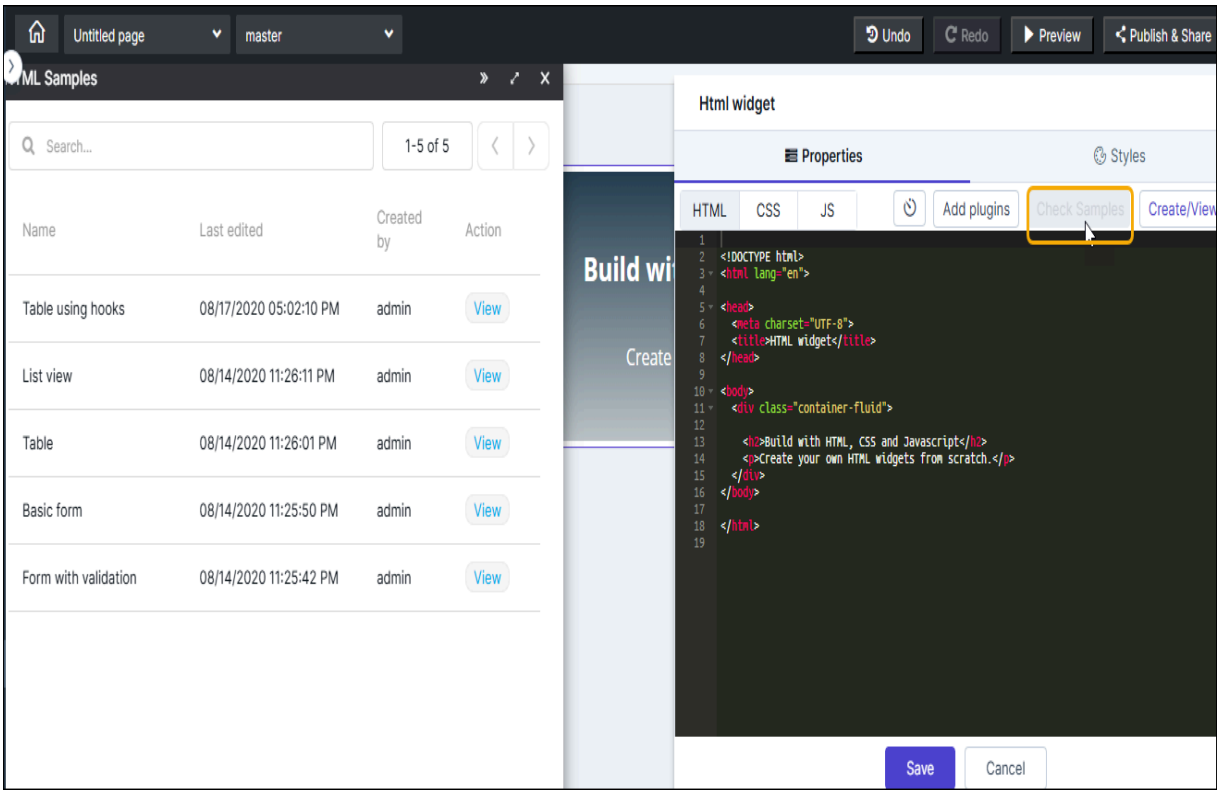
To create a HTML page using sample script:

1. In the **HTML widget** window, click **Check Samples**.



A list of **HTML Samples** is displayed on the left side of the screen.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. To see the details of a sample script, in the **HTML Samples** window, click **View** next to the sample script you wish to view. For example, Basic form.
   The details of the sample script as displayed in the same window.
3. To see a preview of the selected HTML page, click **View Output**.
   The selected HTML page is displayed in the **HTML Samples** window.
4. To use this script to build the HTML page, click **Use Sample**.
   The selected sample script is displayed in the **Html widget** window.
5. To save your selection, click **Save**.

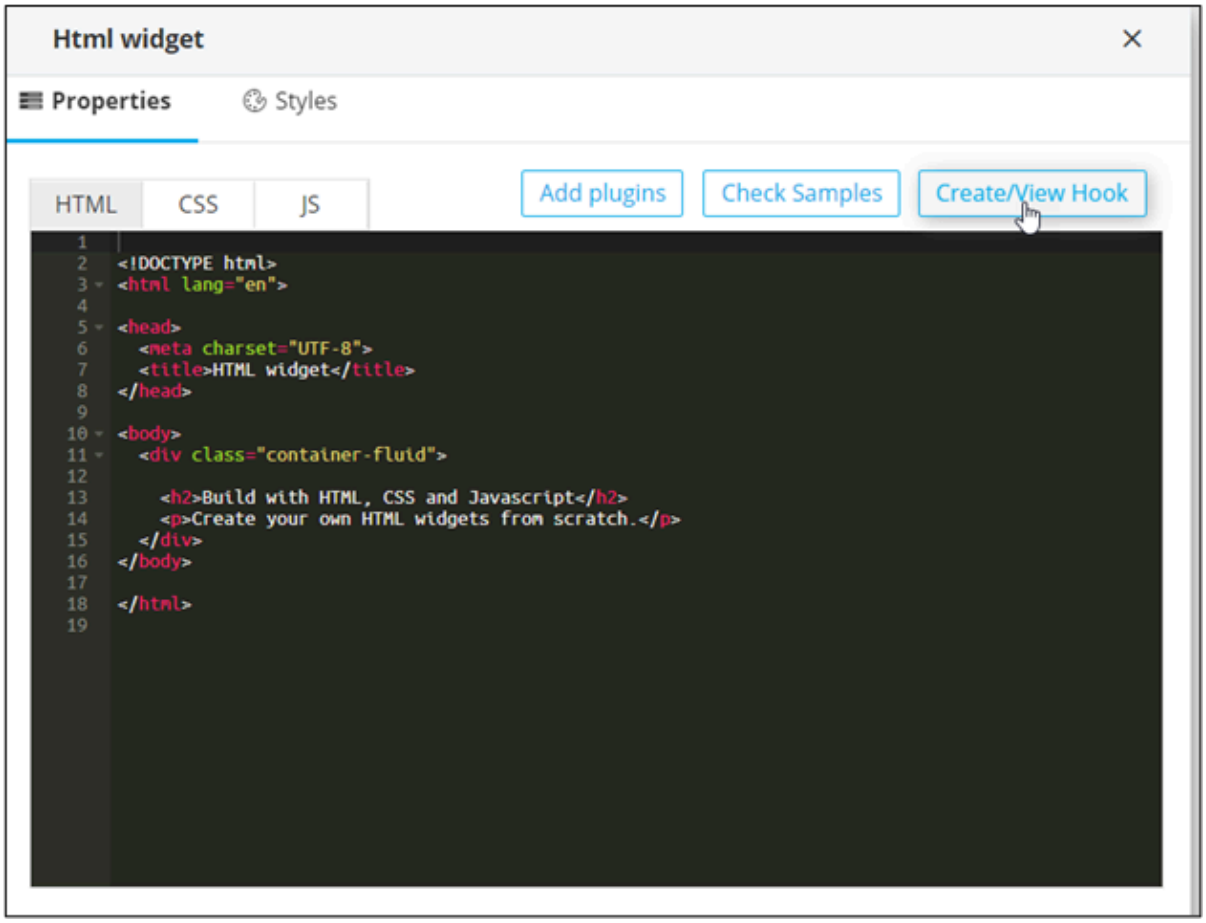
Creating custom HTML pages using Hook Inventory

Fetch information from a data source using Hooks. There is a provision to select pre-existing hooks or create new hooks in the Hooks Inventory and use them to build customized HTML pages.

To use Hooks for building an HTML page:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. In the **Html widget** window, under **Properties**, click **Create/View Hook**.



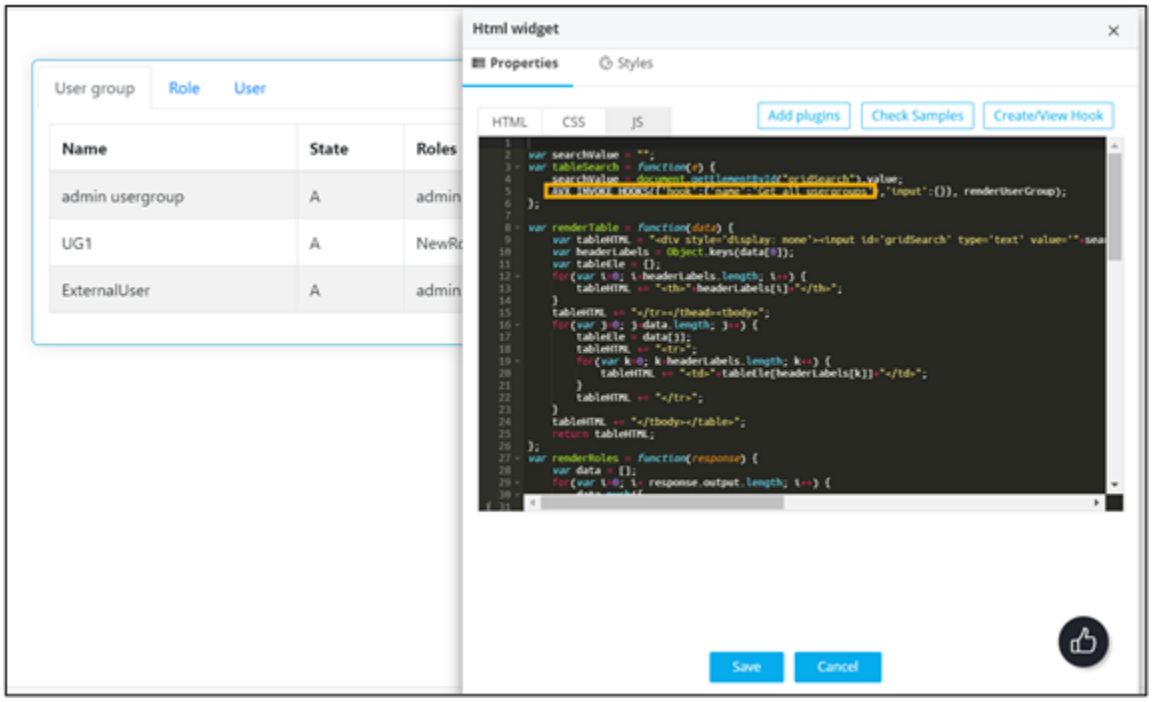
The Hooks Inventory opens in a new tab.

Note: For more information on **Hooks**, refer to the Automation User Guide.

2. To build your customized HTML page, you can either use a preexisting hook from the Hook Inventory or create a new hook.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

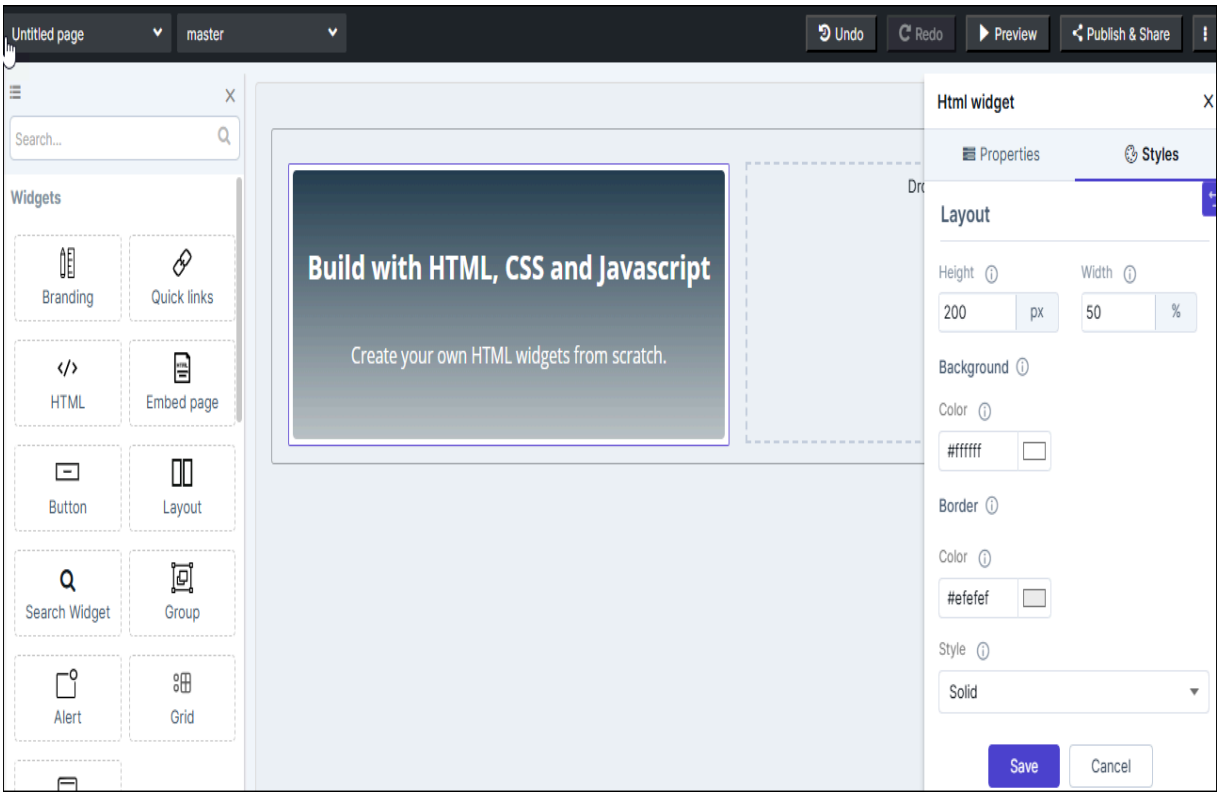
For example, Table using hooks created using **Get all usergroups** hook.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring HTML Widget - Styles

1. For more options to customize the HTML widget, in the **HTML widget** window, click **Styles**.



Field descriptions for Styles

Field	Description
Height	Allows you to increase/decrease the height of the widget.
Width	Allows you to increase/decrease the width of the widget.
Background - Color	Allows you to define the **Background color** for the HTML widget.
Border	Allows you to define the • **Border color** - Select the color of the HTML widget's border. • **Border style** - Select the border style for the HTML widget from the options available in the dropdown for example - dashed, dotted, etc. • **Border width** - Select the thickness of the HTML widget's border.

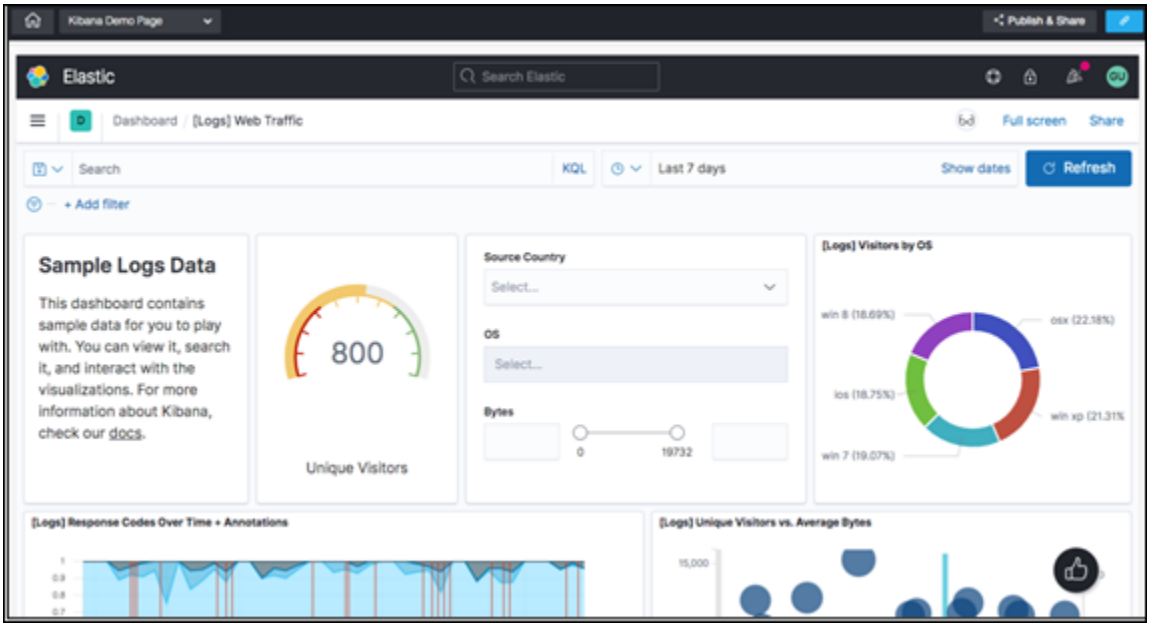
2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

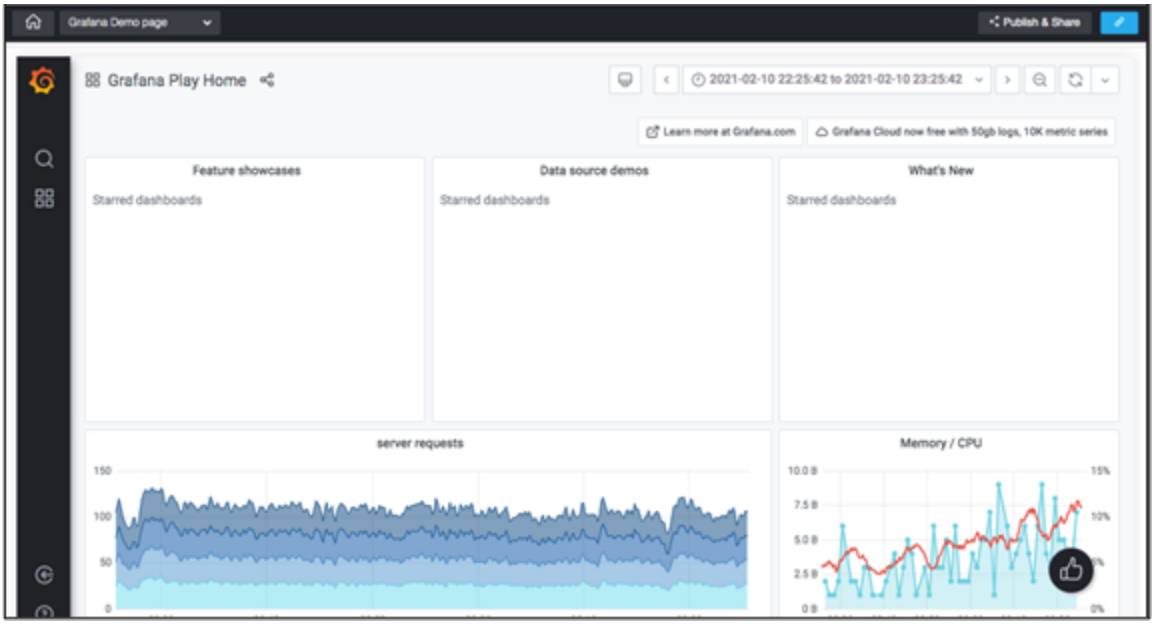
Embed page

The Embed Page widget is a tool that allows users to link or display another entire web page within their page. This feature enables users to include relevant external or internal content seamlessly, enriching their page with additional information. You can embed an entire web page inside a single widget in your service catalogs.

For example, embed a Kibana visualization dashboard as show below.



Or, embed a Grafana visualization dashboard below.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Embedding a page - Properties

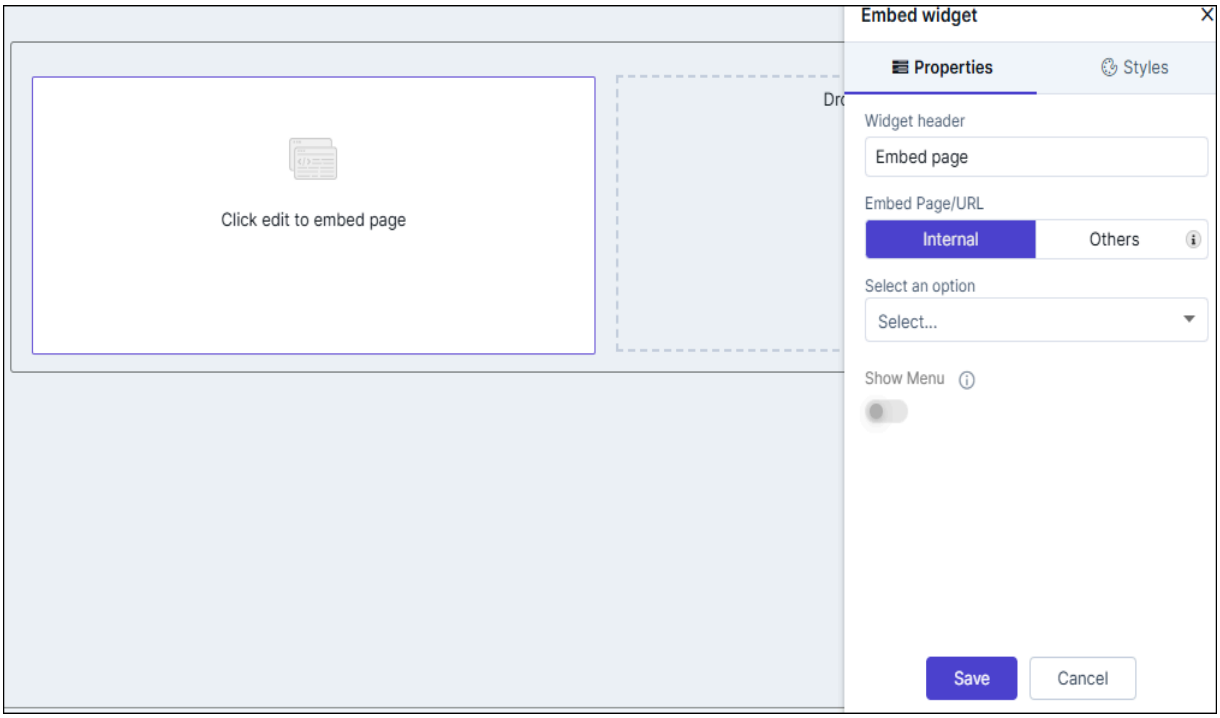
You can either choose from a predefined list of internal pages or embed an external page into the widget. Cross domain data exchange must be possible to embed an external page into your catalog.

To embed a page:

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Embed page** widget to any blue highlighted space on the page.
2. Hover your mouse over the widget and click **(Edit)** icon.

The Embed widget pop-up window opens with the **Properties** tab open by default.

3. In the **Embed widget** window, under **Properties**, enter or select the required field information.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

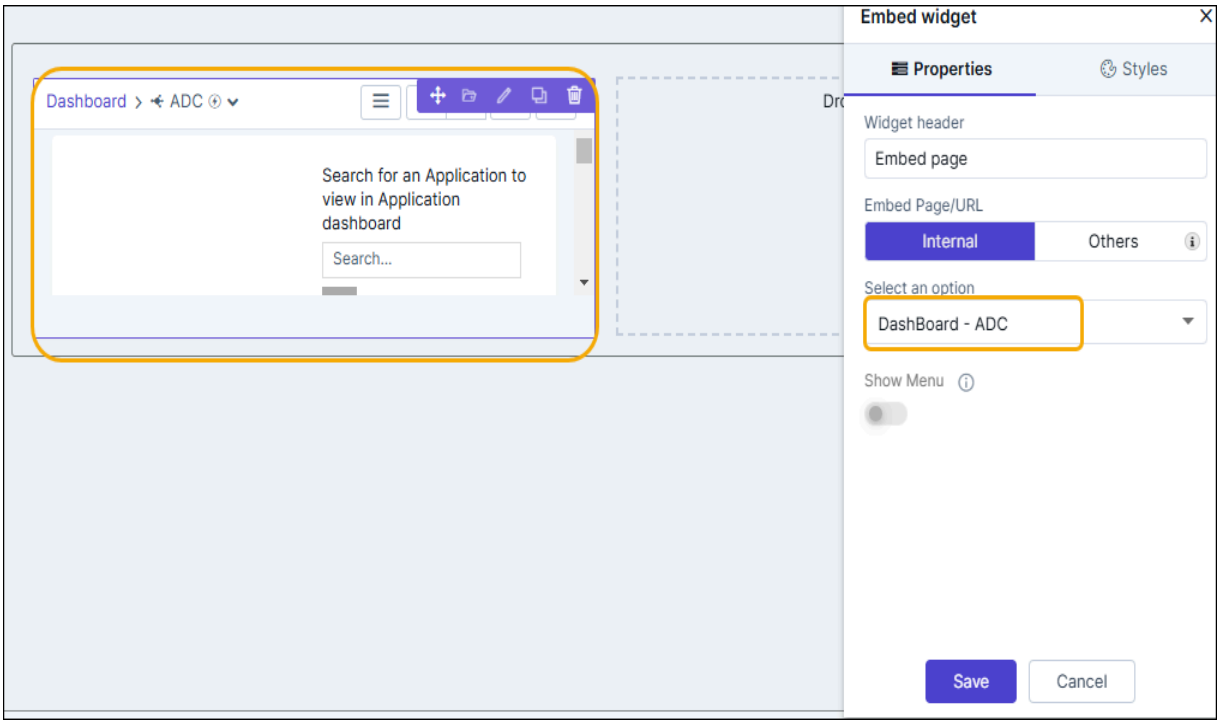
Field descriptions for Properties

Field	Description
Widget Header	Enter a suitable widget header
Embed Page/URL	You have two options for embedding a page in the widget: • **Internal**: Add an internal link from within AppViewX pages. • **External**: Users can embed cross origin enabled pages here.

4. Click **Save**.

Embedding an Internal Page

1. In the **Embed widget** window, under **Internal**, select an option from the drop down list. For example, **Dashboard - ADC**.



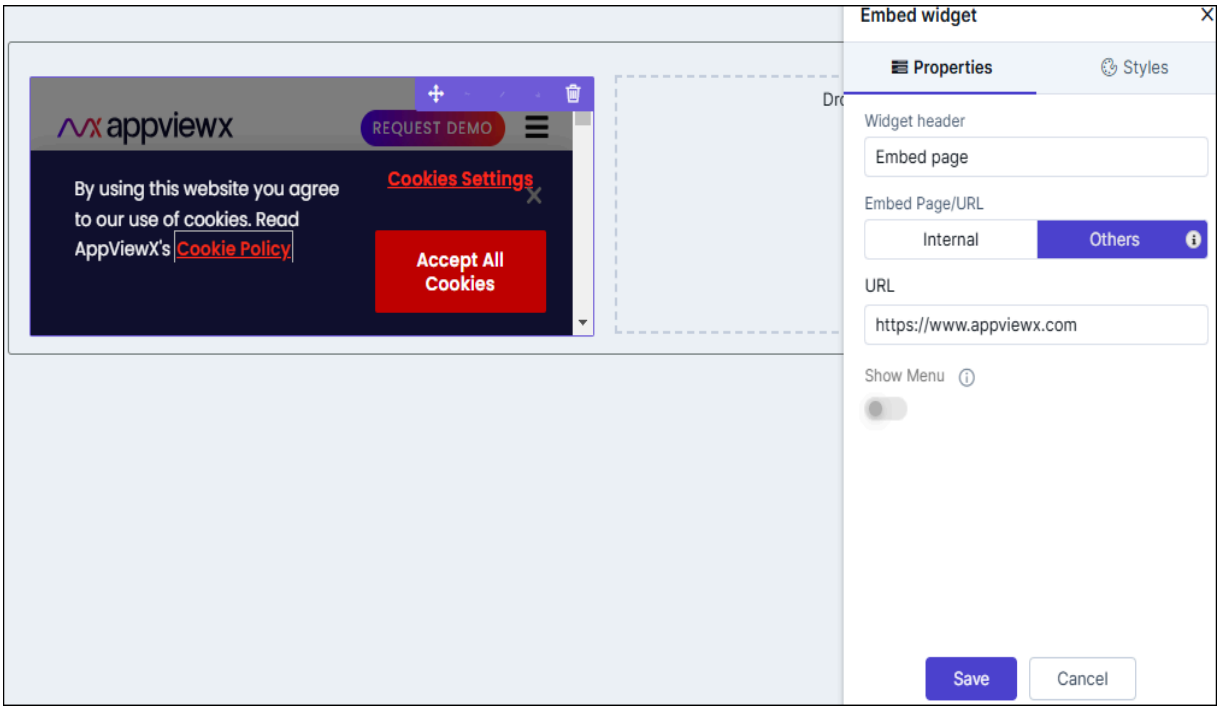
2. To embed this page, click **Save**.

   The selected page is embedded and displayed in your service catalog.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Embedding an External Page

1. In the **Embed widget** window, under **Properties**, click **Others**.



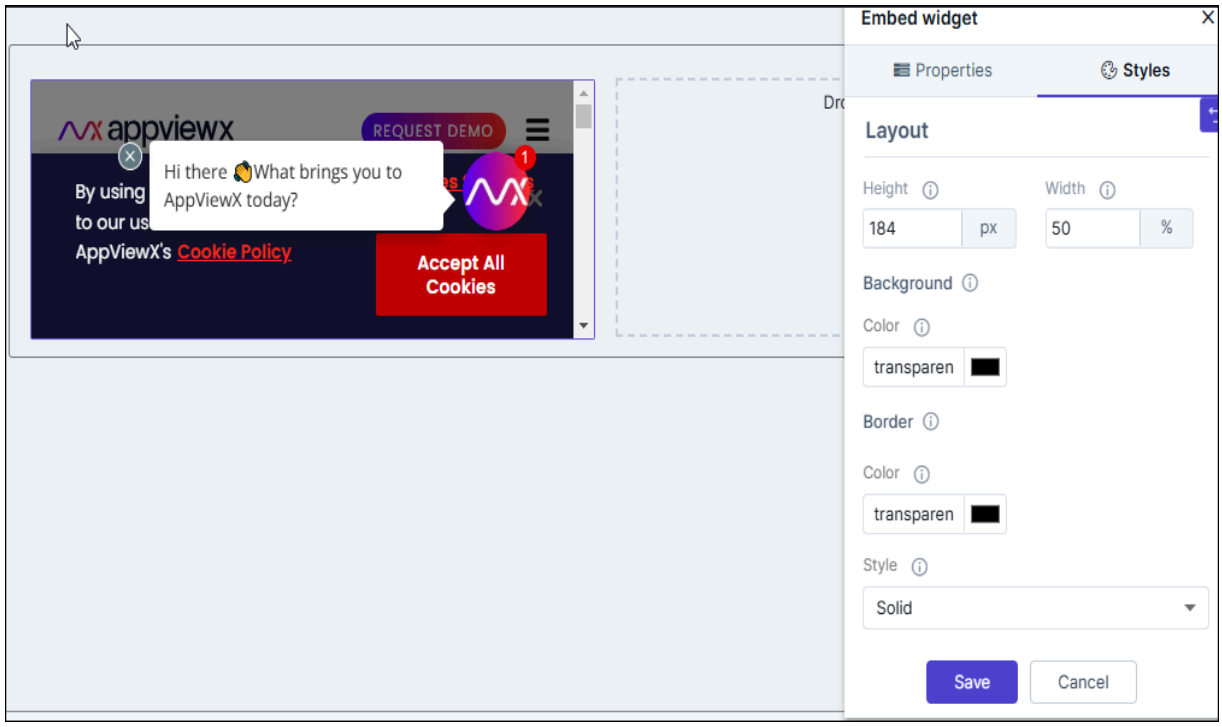
2. Enter the URL of the page that is to be embedded.
3. To embed the page, click **Save**.

   The page is embedded in your service catalog.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Embedding a Page - Styles

1. For more options to customize the **Embed widget**, in the **Embed widget** window, click **Styles**.

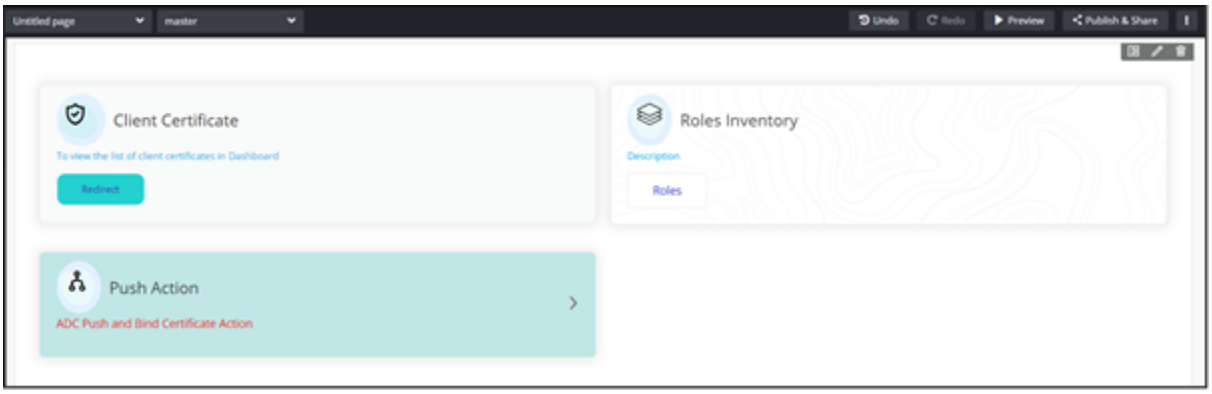


2. Under **Sizing**, you can increase/decrease the size of the widget.
3. To save your settings, click **Save**.
4. To restore the settings under **Styles** to default, click **(Restore)** icon.

Button (Action Widget)

The Button widget is a versatile component that allows users to add a single URL link with various customizable configurations to their pages. Unlike the Quick Links widget, which supports multiple links within a single widget, the Button widget focuses on providing a clear call-to-action with a single destination URL. You can add buttons to your service catalog to trigger specific “call to action”.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



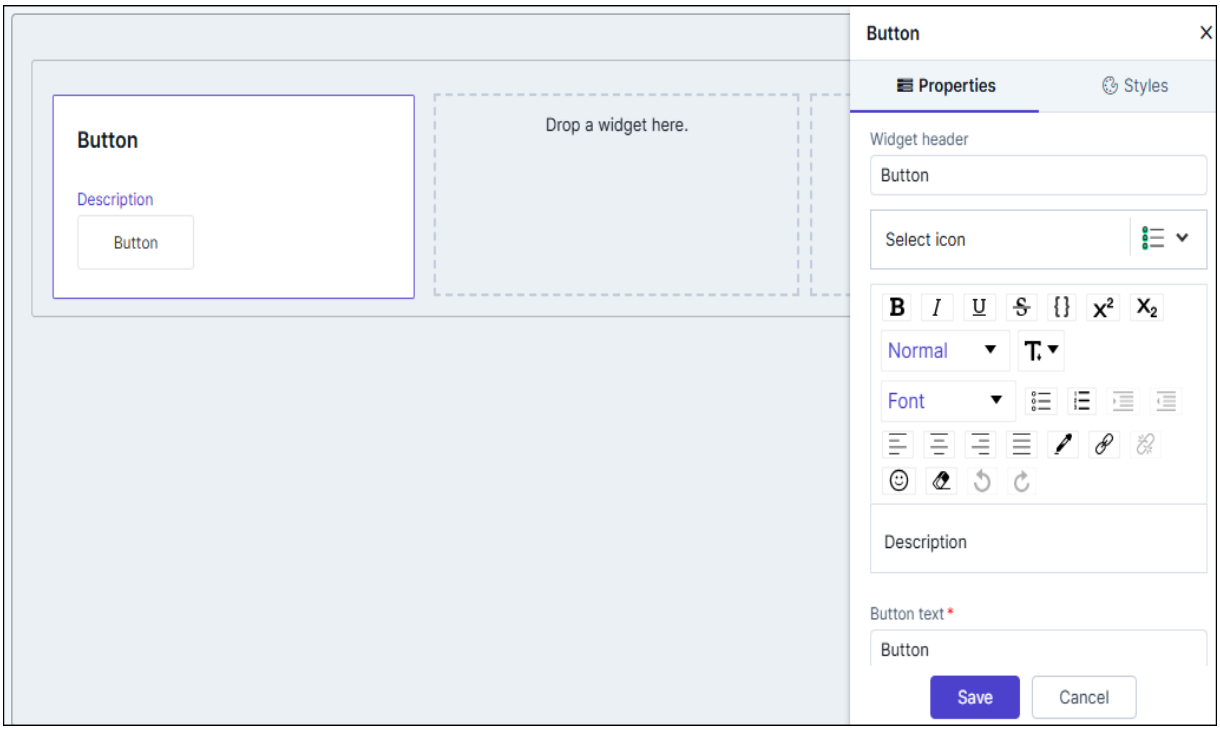
Configuring the Button/Action Widget - Properties

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Button** widget to any blue highlighted space on the page.
2. To configure the button/action widget, hover your mouse over the widget and click **(Edit)** icon.

The **Button** pop-up window opens.

3. In the Button pop-up window, under **Properties**, enter or select the required fields.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Properties

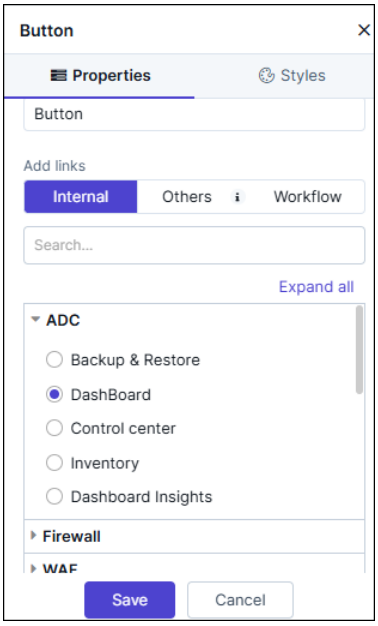
Field	Description
Widget Header	Allows you to enter the widget header name.
Select icon	Allows you to select the icon for the widget from the options available in the dropdown.
Description	Allows you to add a description for the action widget.
Button text	Allows you to add the text that is displayed on the button.
Add links	Allows you to add links to the action widget. The following options are available here: • **Internal**: Add an internal from within AppViewX pages. • **Others**: Add links to other pages within AppViewX or embed external cross origin enabled pages. • **Workflow**: Add a workflow link which redirects to the workflow execution page.

4. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Adding an Internal Action Widget Link

1. To link an internal AppviewX page to the action widget, under **Add links**, click **Internal**.
2. Select the required internal link from the list.



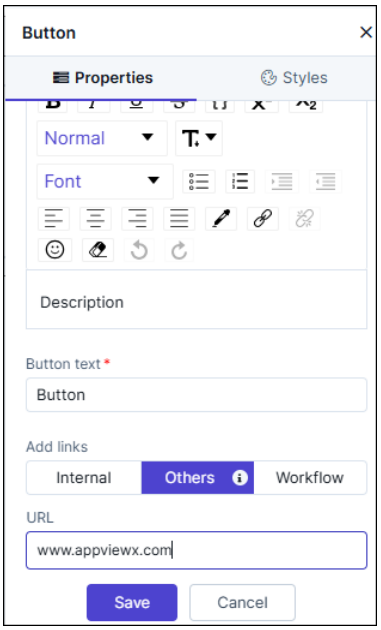
3. Click **Save**.

   Clicking on the button will redirect to the selected link in a new tab.

Adding an Others/External Widget Link

1. To embed an external link or a cross origin enabled page URL into the Action widget, under **Add links**, click **Others**.
2. Enter the **Name** and **URL** for the link.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

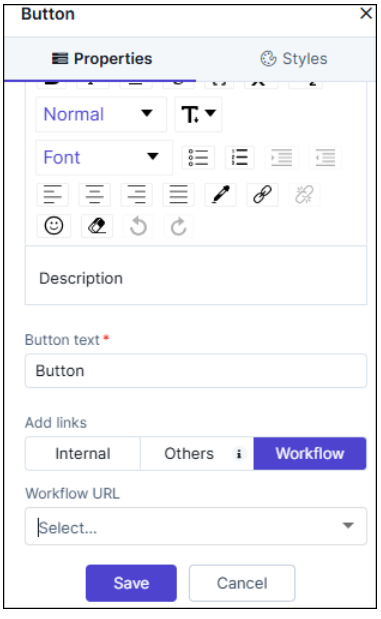


3. Click **Save**.

   Clicking on the button will redirect to the page as per the configured URL.

Adding a Workflow Link to the Widget

1. To add a link to redirect to the Workflow Request execution page for the selected workflow, under **Add links**, click **Workflow**.
2. Enter the link **Name** and select the **Workflow URL** from the dropdown.

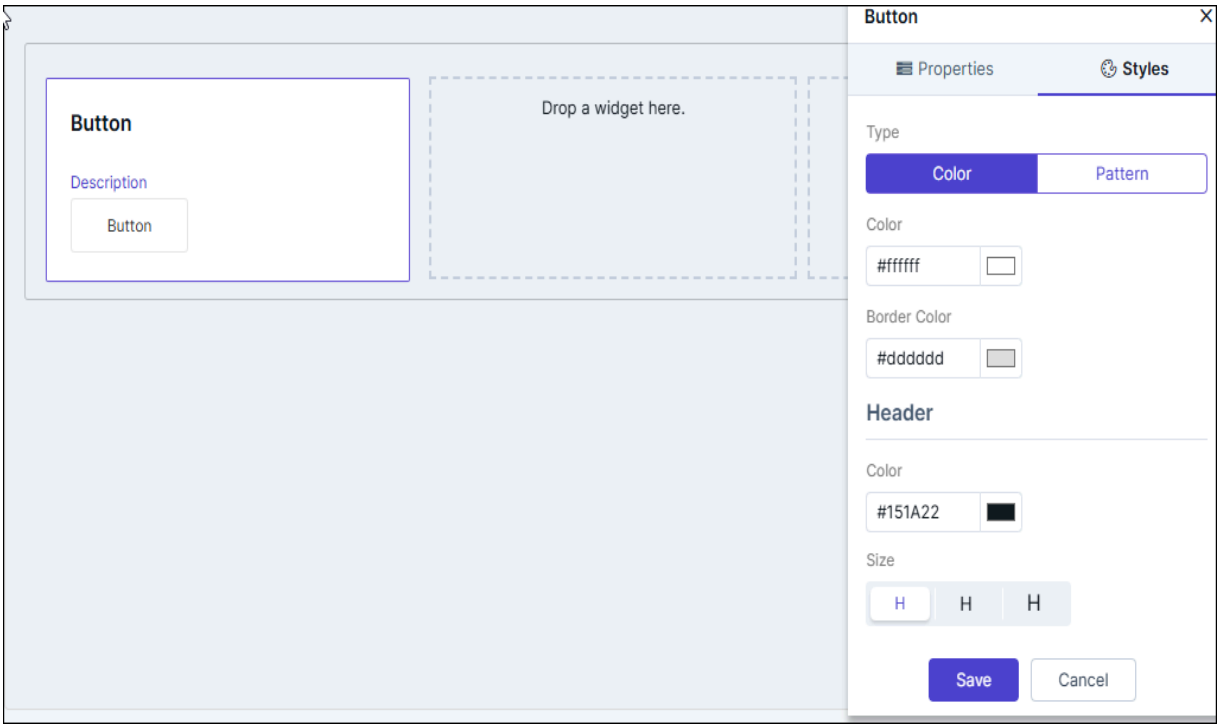


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. Once you have selected the required workflow URL to the action widget, click **Save**.
   Clicking on the button will redirect to the workflow request execution page for the configured workflow URL.

Configuring the Button/Action Widget - Styles

1. For more options to customize the action widget, in the **Button** window, click **Styles**.



Field descriptions for Styles

Option	Description
Sizing	Allows you to increase/decrease the width of the action widget.
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, unselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.
Widget Description	The following options are available here: • **Description text Color**: Allows you to change the color of the label text. • **Description Text Size**: Allows you to select the size of the label text. • **Description Text Underline**: To underline the description text, select the **Description Text Underline** checkbox.
Widget Type	Allows you to enable/disable the button in the action widget. • To enable the button select the **Enable button** checkbox. • To disable the button, deselect the **Enable button** checkbox. The action widget is displayed in a card view. Click on the widget to redirect to the page that is linked to the widget.
Button Background	This feature is available only when the button is enabled. The following options are available here: • **Button Corners**: Allows you to define the curvature of the button corners. The higher the number, the corners will be curved more. • **Button Alignment**: Allows you to define the placement of the button in the action widget - left, center, and right. • **Widget Background Color**: Allows you to select the background color of the widget from the color palette. • **Widget Border Color**: Allows you to select the color of the widget border from the color palette.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
Button Text	This feature is available only when the button is enabled. The following options are available here: • **Button Text Alignment**: Allows you to define the placement of the button text inside the button - left, center, and right. • **Button Text color**: Allows you to select the button text color from the color palette. • **Text Size**: Allows you to select the size of the text displayed on the button.
Icon Style	The following options are available here: • **Icon Color** - Select the Icon Color from the available options. • **Icon Background** - Select the Icon Background from the available options. • **Icon Background Color** - Select the Icon Background Color from the available options.

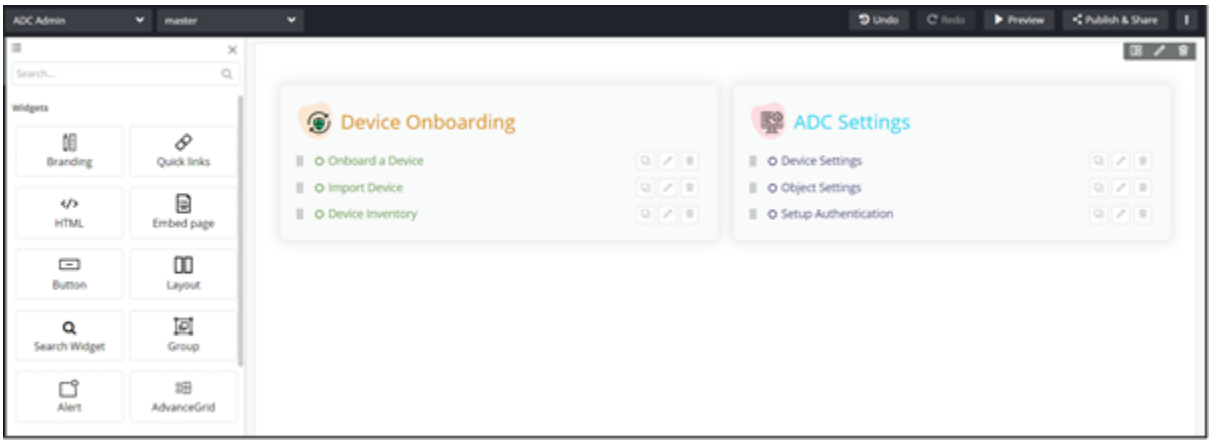
2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Layout

The Layout widget is a versatile tool that allows users to create a structured layout with two empty cells. This widget acts as a parent container for other widgets, providing users with a flexible canvas to arrange and organize their content.

You can customize the layout of your service catalog(s) in the following ways:

- Separate a row into two columns at a time
- Insert more layouts within a layout for additional columns.

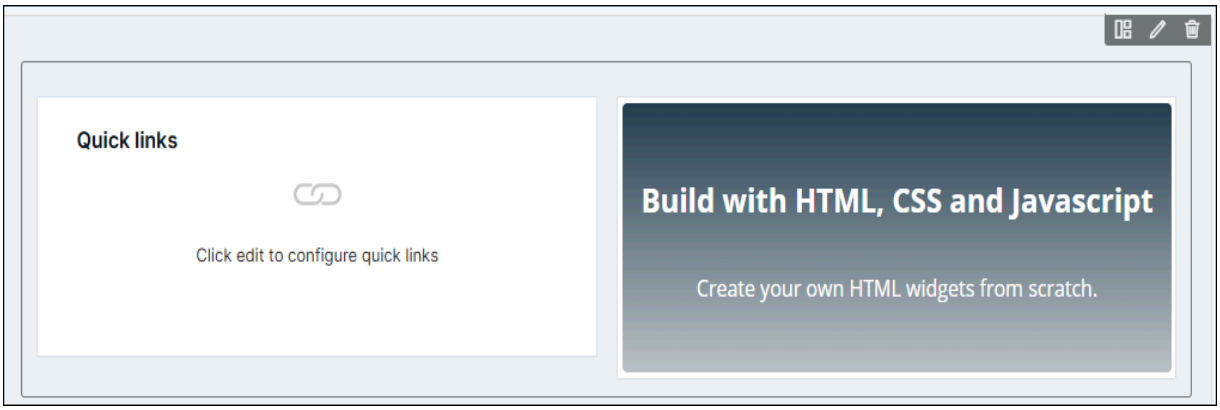


Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Adding a Layout Component

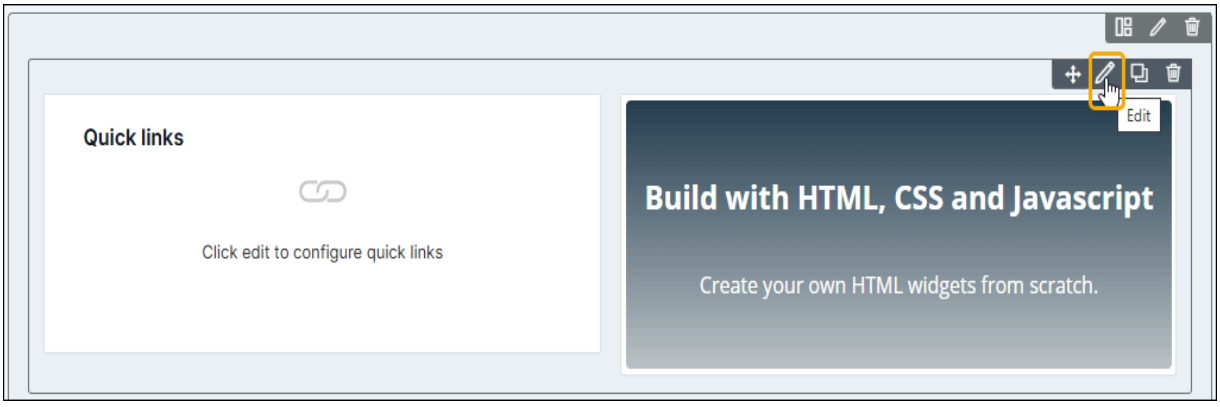
To insert a Layout component to your page:

1. Drag and drop the **Layout** component to any blue highlighted space on the page.
2. Drag and drop one or more component(s) into this **Layout** widget as per your requirement.



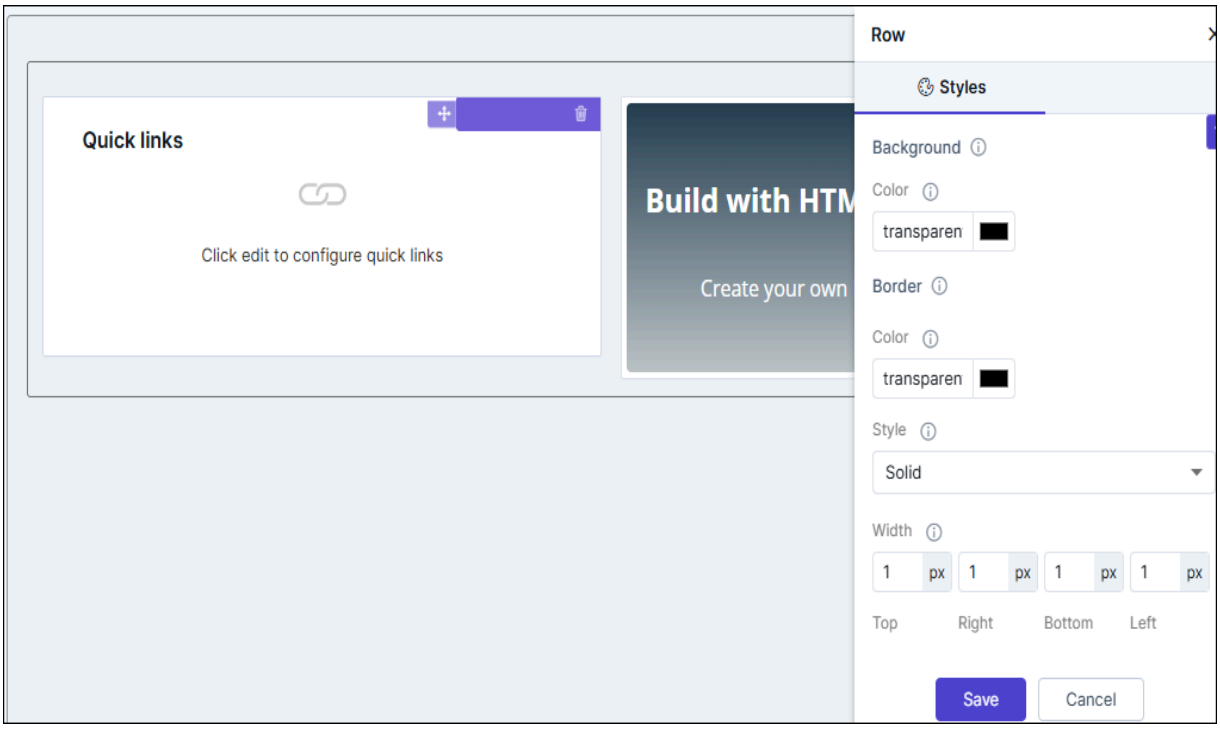
Configuring the Layout Component

1. To configure/edit the Layout component, hover your mouse over the widget and click **(Edit)** icon.



The **Row** pop-up window opens with the **Styles** tab.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Styles

Option	Description
Background	Allows you to define the **Background color** for the Layout widget.
Border	Allows you to define the • **Border color** - Select the color of the layout border. • **Border style** - Select the border style for the layout section from the options available in the dropdown. • **Border width** - Select the thickness of the layout border.

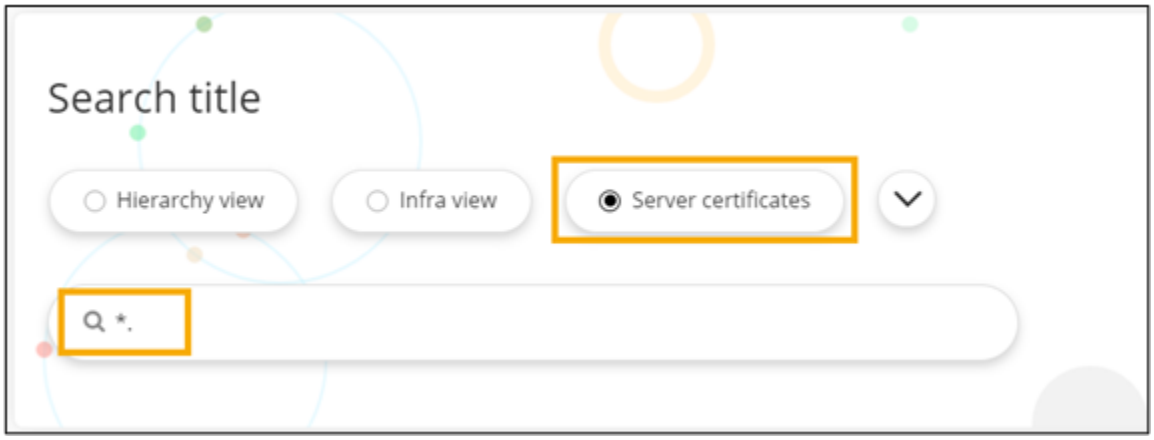
2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

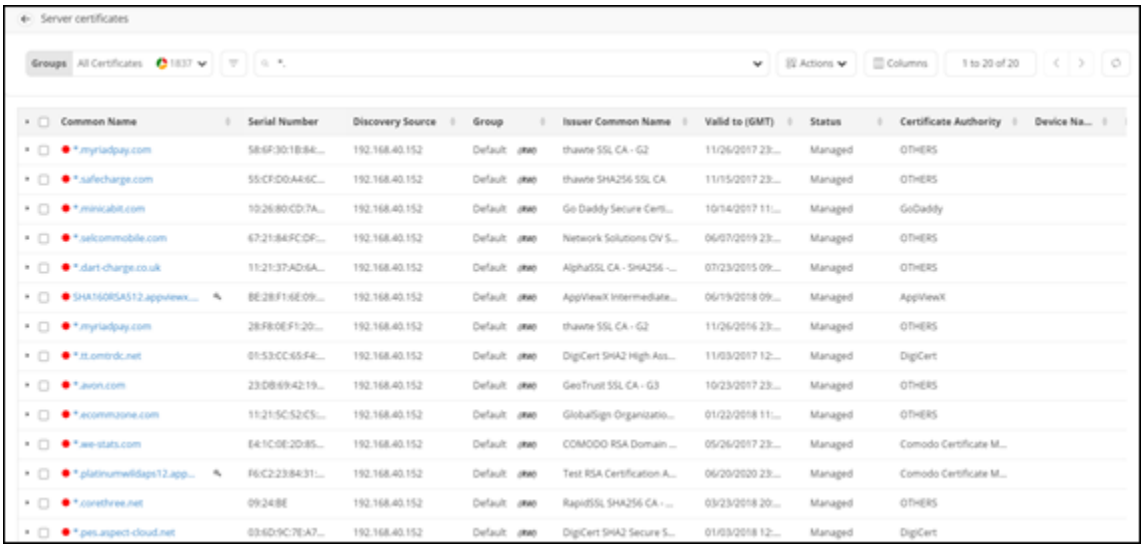
Search Widget

The Search widget is a tool that enables users to search and filter specific content from the entire application. With its built-in search configurations, users can easily locate server certificates, client certificates, and other relevant information within the application.

For example, search for certificates by typing *. in the search field to get a list of all server certificates.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

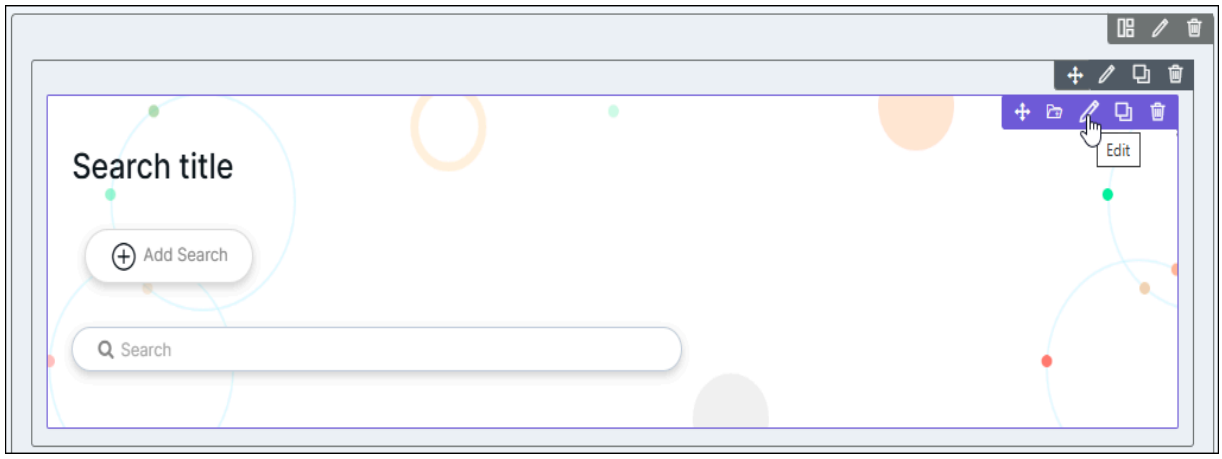




Configuring the Search Widget - Properties

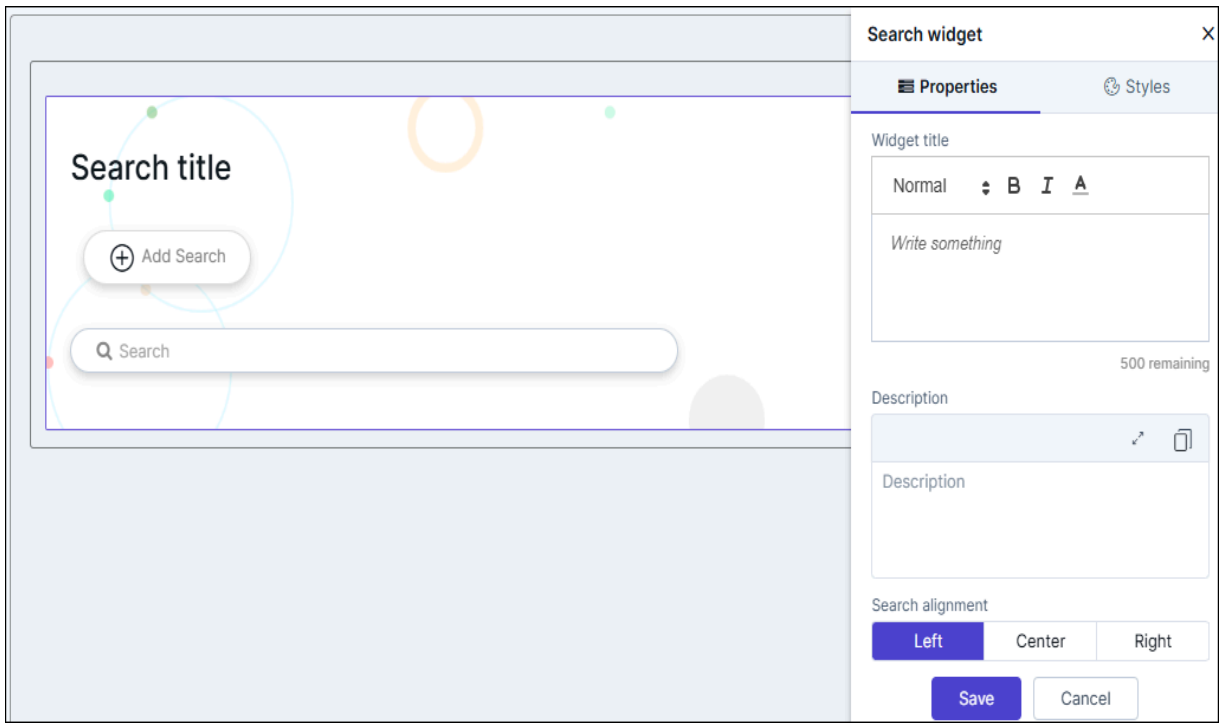
1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Search** widget to any blue highlighted space on the page.
2. To configure/edit the Search Widget component, hover your mouse over the widget and click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Tip: To configure/edit the Search Widget component, you can also click **Add Search** in the Search widget.

The **Search widget** window opens with the **Properties** tab open by default.



3. In the **Search widget** window, under **Properties**, enter or select the required field information. The following table describes the various fields in this tab:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Properties

Field	Description
Widget title	Provide a suitable title for the Search widget. **Note:** You have the option to set the font size, color, and thickness in the **Widget title** editor box.
Description	Allows you to add a **Description** for the search widget.
Search Alignment	Allows users to align the search widget according to their preferred layout and design.
Configure Widget	Select the search parameters that would appear in the search widget by checking the box. **Note:** You have the option to edit the search widgets, add descriptions and tooltips by clicking **(Edit)** icon. For more information, click here.
Widget Background	Choose a widget background from the available options. To keep a transparent background, enable the toggle button **Enable transparent background**.

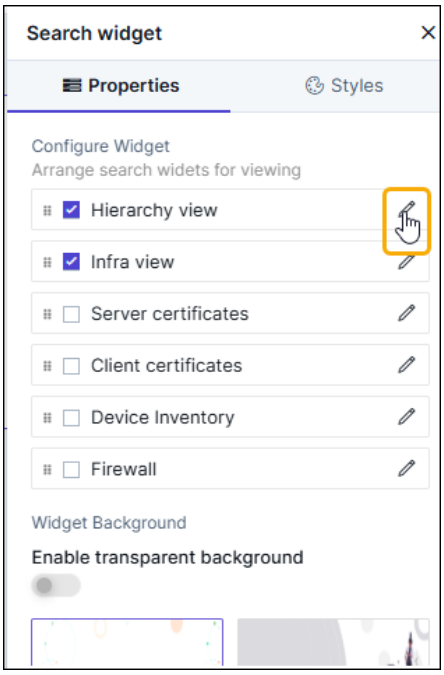
4. Click **Save**.

Editing the Search Widget

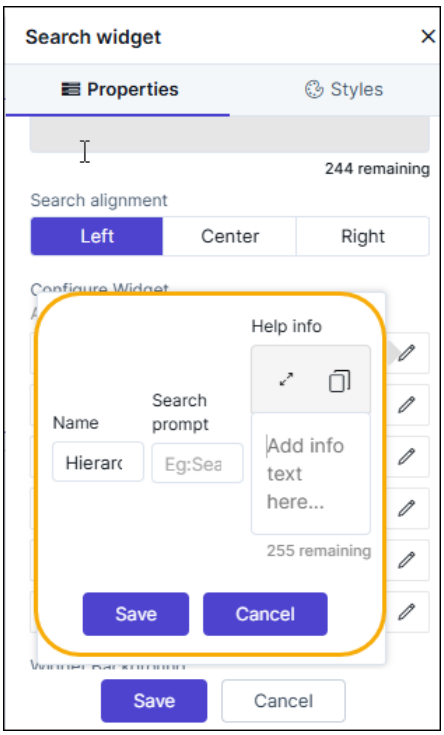
To edit a specific search widget:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. In the **Search widget** window, under **Properties**, click **(Edit)** icon next to the widget to be edited.



2. Enter all the field information.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

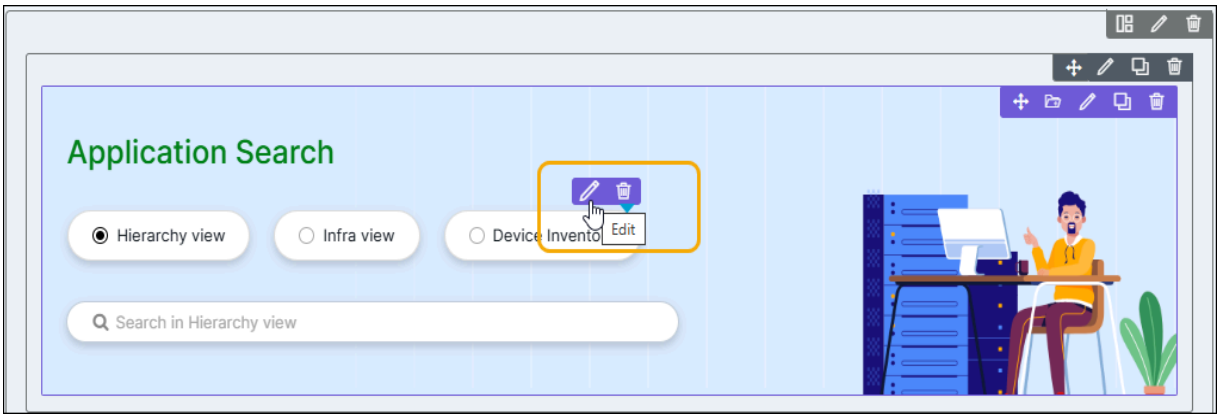
Field descriptions for Properties

Field	Description
Name	Allows you to change the name of the search widget.
Search prompt	Allow you to add a search prompt in the search bar when a particular search widget is selected.
Help info	Allows you to add a help text to the search widget.

3. Click **Save**.

   The search prompt and help info is updated in the search widget.

> **Note:** You can also edit the search widget, by hovering your mouse over the search widget and clicking **(Edit)** icon.



Deleting a single search widget

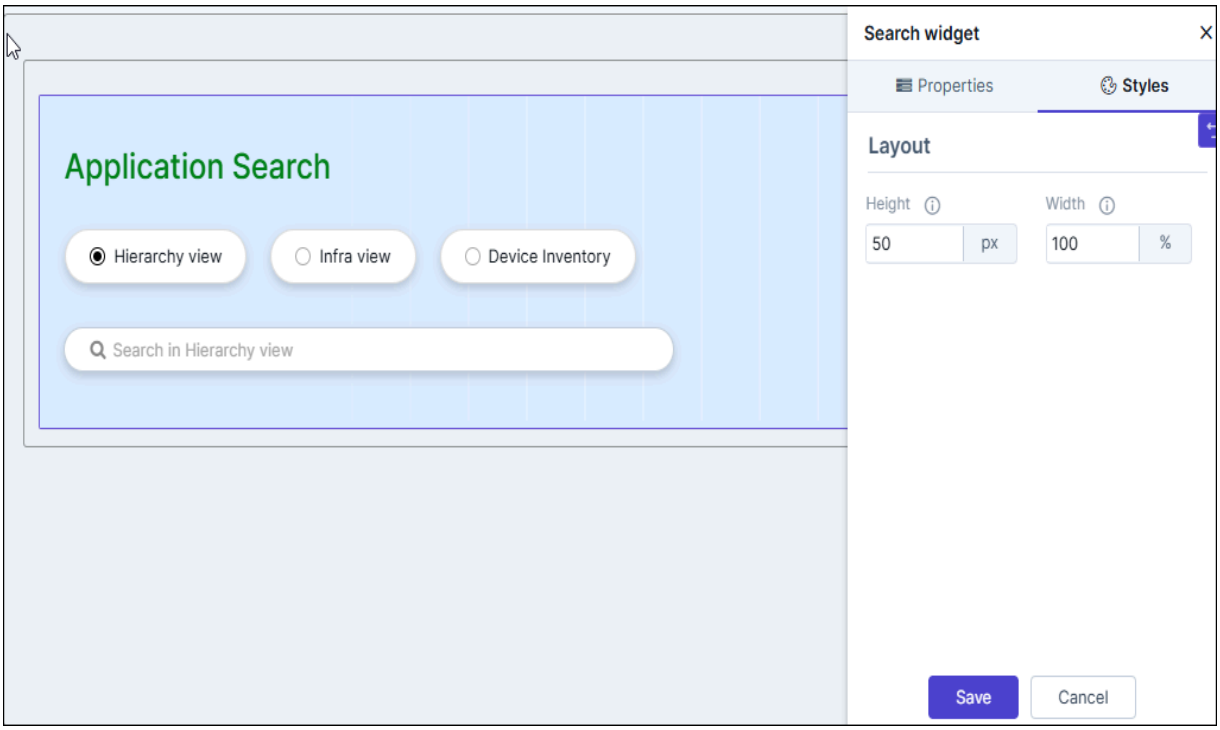
1. Hover your mouse over the search widget to be deleted.
2. Click **(Delete)** icon.

   The selected search widget is deleted.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Search Widget - Styles

1. For more options to customize the **Search Widget**, in the **Search Widget** window, click **Styles**.

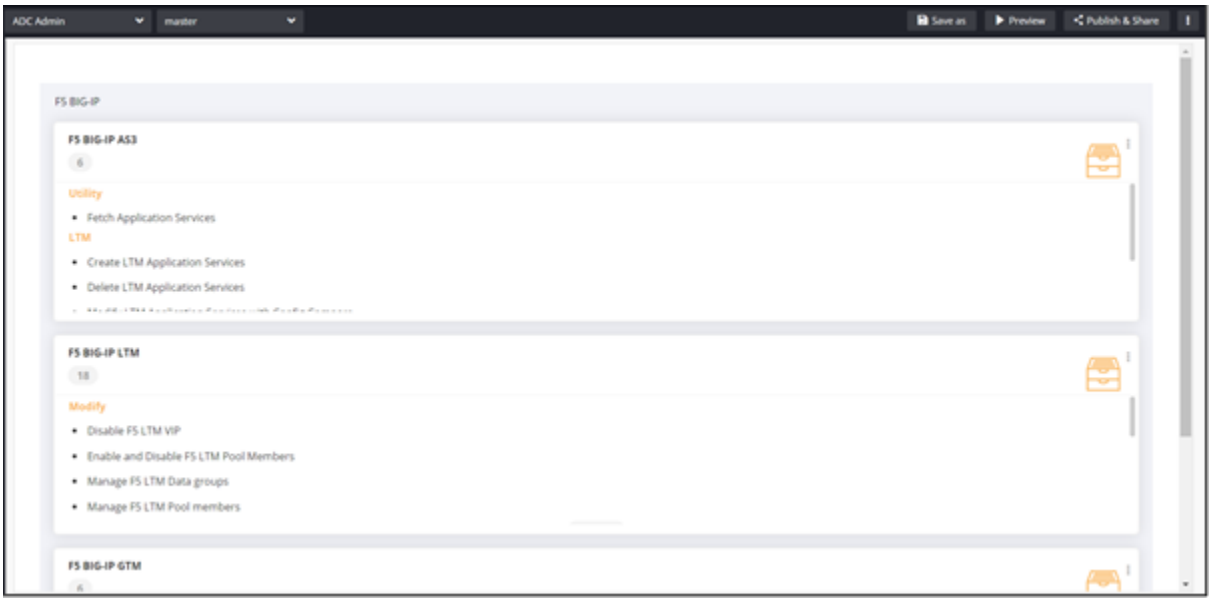


2. Under **Sizing**, you can increase/decrease the height and width of the search widget.
3. To save your settings, click **Save**.
4. To restore the settings under **Styles** to default, click **(Restore)** icon.

Group

The Group widget is a tool that facilitates the grouping of various components within the page. This widget allows users to create logical clusters of related components, providing a cohesive and organized presentation.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

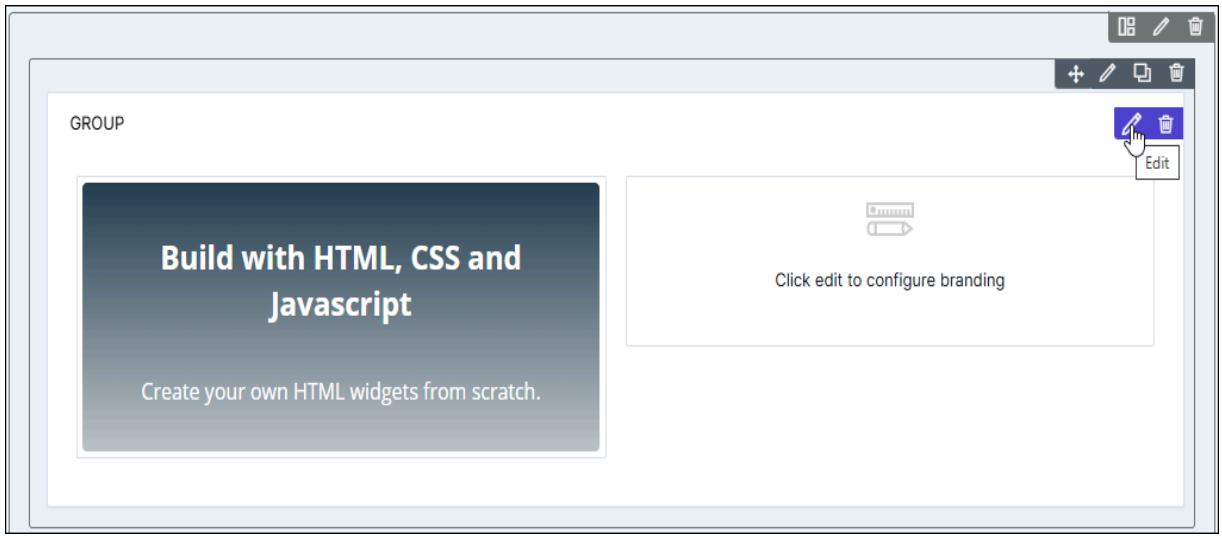


Configuring the Group Widget - Properties

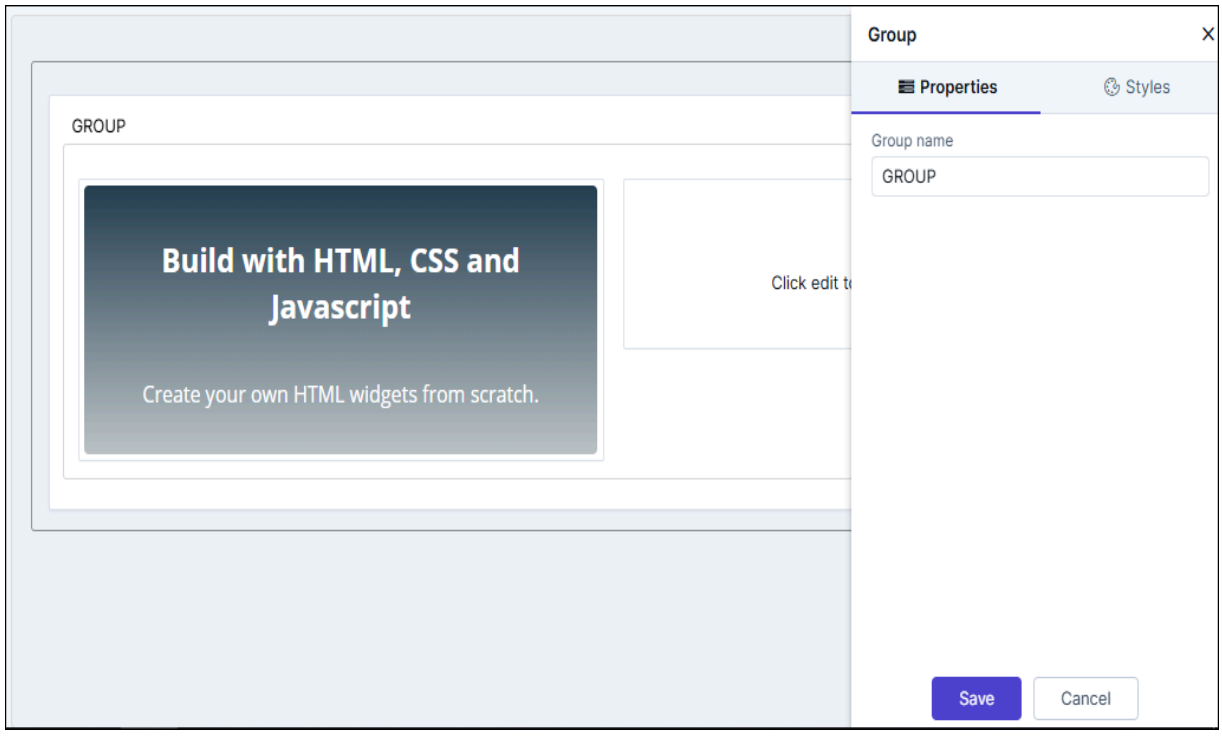
1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Group** widget to any blue highlighted space on the page.

2. Drag and drop related widgets into this **Group** component to view them all in one place.
3. To configure/edit the Group component, click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **Group** pop-up window opens with the **Properties** tab open by default.

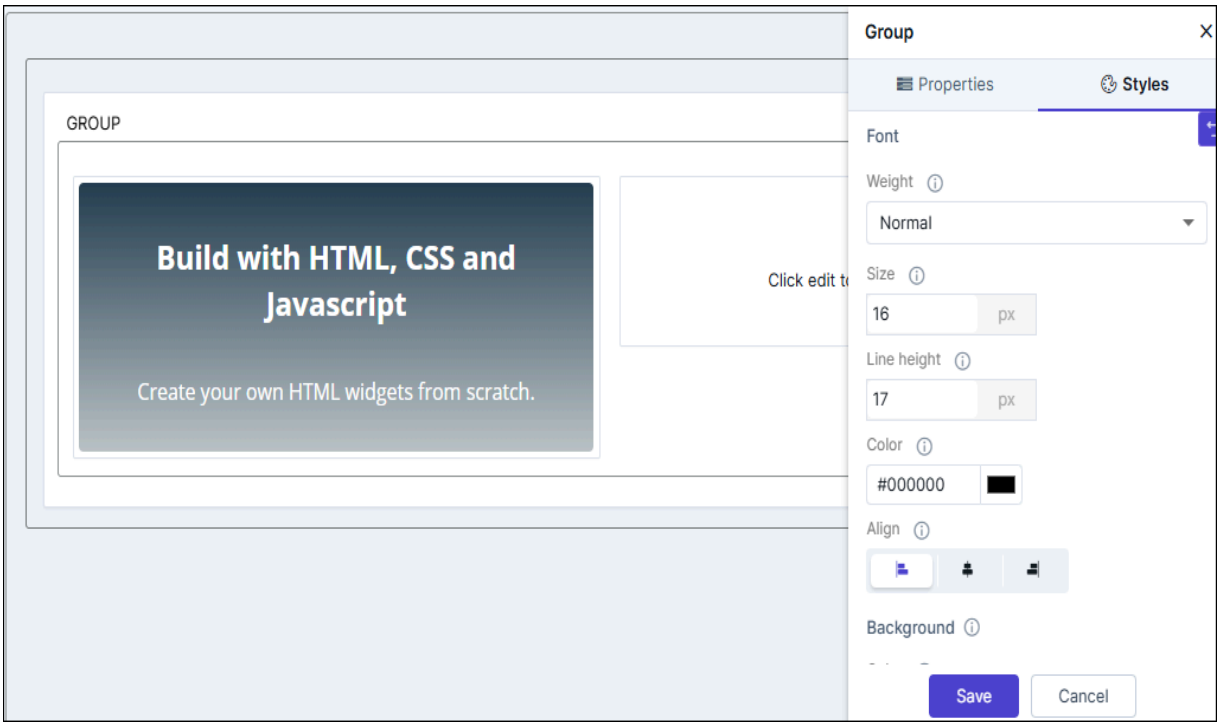


4. In the **Group** window, under **Properties**, you can change the **Group name**.
5. Click **Save**.
6. To configure individual widgets within the group, hover your mouse over the specific widget and click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Group Widget - Styles

1. For more options to customize the **Group** component, in the **Group** window, click **Styles**.



Field descriptions for Styles

Option	Description
Weight	The **Weight** dropdown list lets you set the font thickness of the field label using one of the following values: • **Default** • **Light** • **Normal** • **Thick**.
Size	Allows you to increase/decrease font **Size** of the field name.
Line height	Allows you to increase/decrease the **Line height** of the field. This means the space between the field and the field border.
Color	Allows you to define the **Color** of the field label.
Align	Allows you to **Align** the field label to the left, center or right side of the field.
Background	Allows you to define the **Background color** for the HTML widget.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
Border	Allows you to define the • **Border color** - Select the color of the layout border. • **Border style** - Select the border style for the layout section from the options available in the dropdown. • **Border width** - Select the thickness of the layout border.

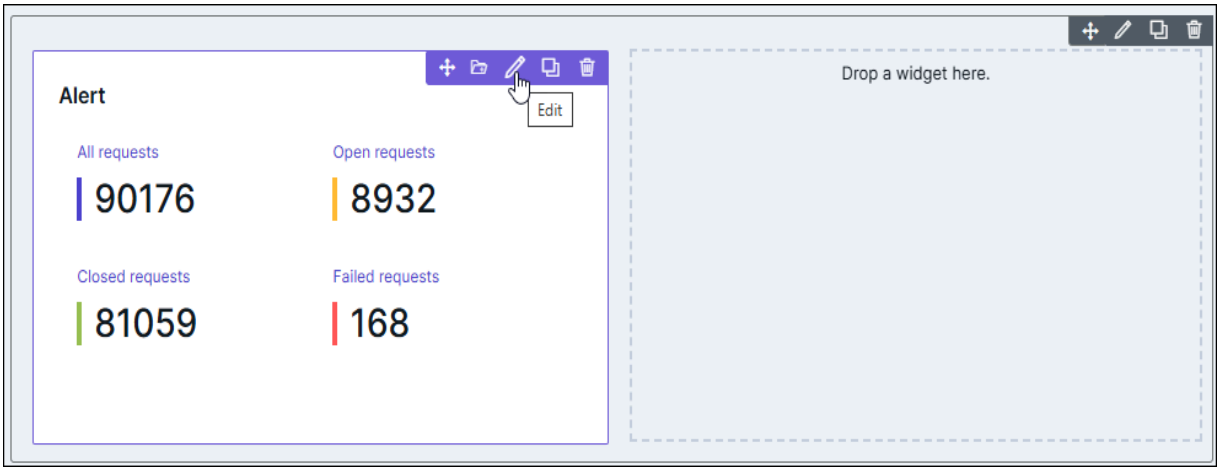
2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Alert

The Alert widget is a tool that displays various visual workflow (VW) requests available in the environment. It provides users with a comprehensive overview of different types of VW requests, facilitating better decision-making and resource allocation. The Alert widget serves as a valuable resource for users to monitor and manage VW requests efficiently, enhancing their ability to oversee the environment effectively.

Configuring the Alert Widget - Properties

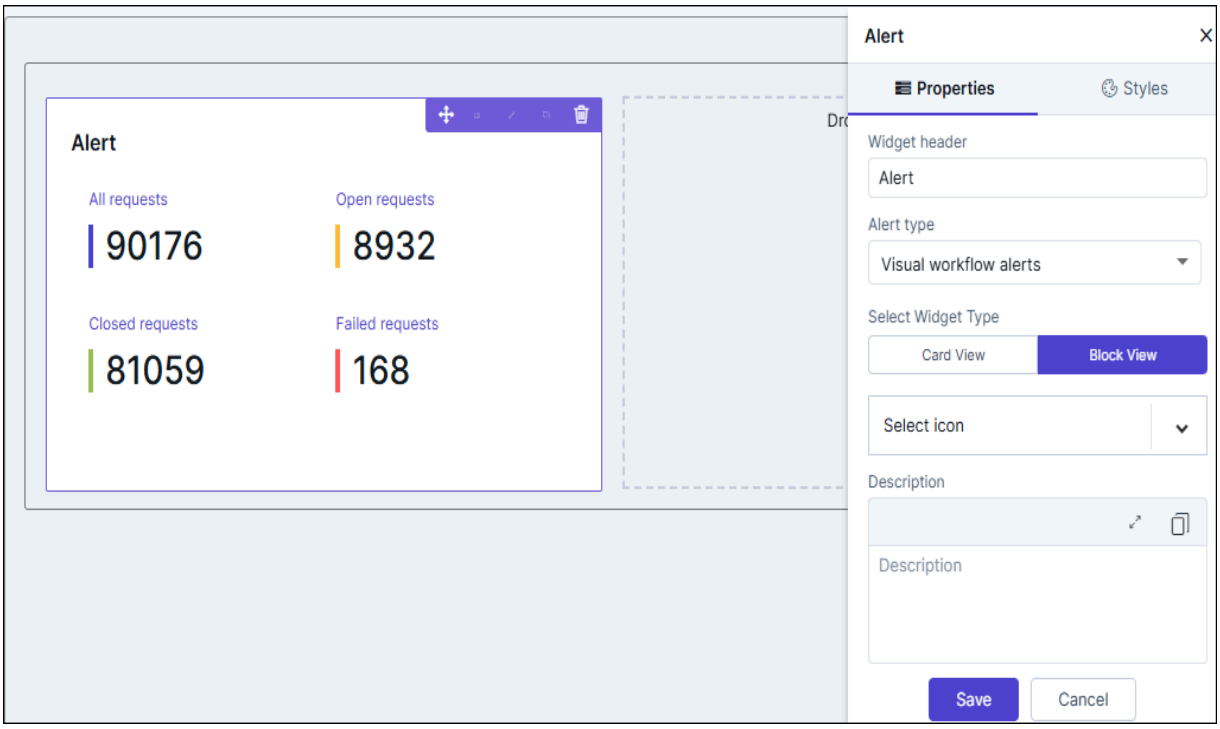
1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Alert** widget to any blue highlighted space on the page.
2. To configure/edit the **Alert** widget, hover your mouse over the widget and click **(Edit)** icon.



The **Alert** widget window opens with the **Properties** tab open by default.

3. In the **Alert** window, under the **Properties** tab, enter or select the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Properties

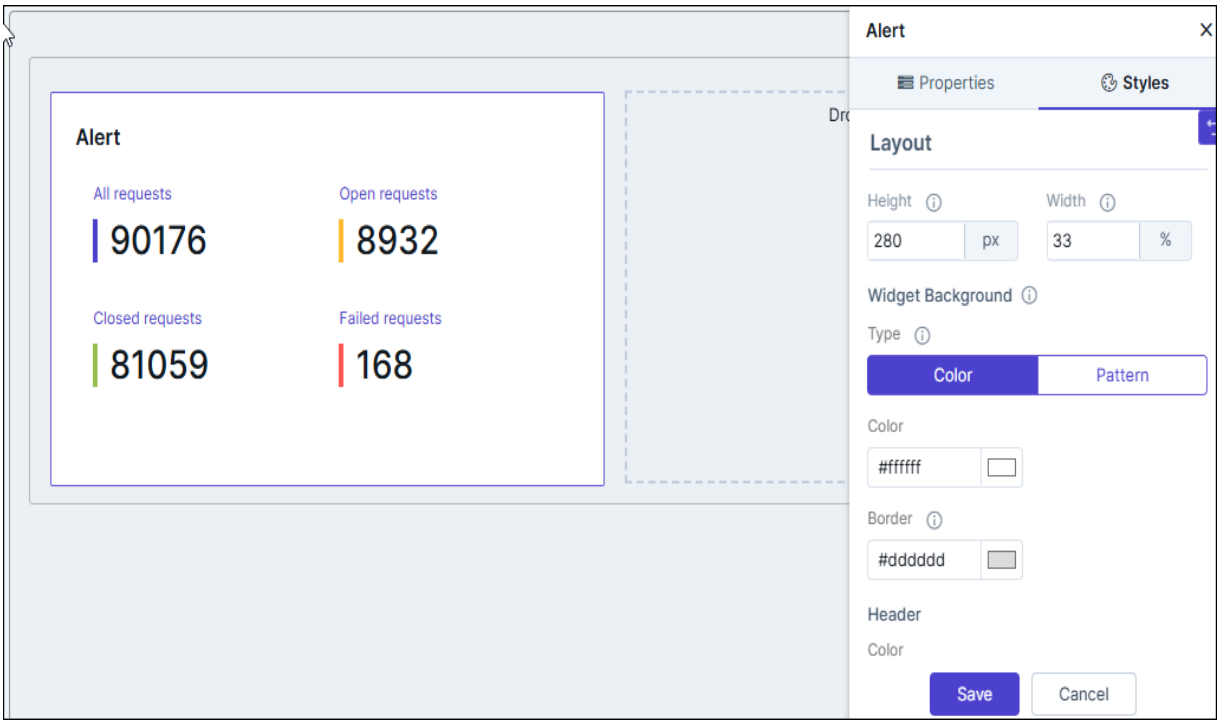
Field	Description
Widget header	Allows you to rename the widget header.
Alert Type	Allows you to select the **Alert Type** from the options available in the dropdown.
Select Widget Type	Allows you to switch between **Card View** and **Block View**. • **Card View:** In this option, the VW requests are displayed in a card layout, providing a concise view with essential information for each request. • **Block View:** In this option, the VW requests are displayed in a block layout, allowing users to view multiple requests in a list format.
Select Widget Icon	Allows you to select an icon for the widget from the options available in the dropdown.
Description	Allows you to add a description for the widget.

4. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Alert widget - Styles

1. For more options to customize the Alert widget, in the **Alert** window, click **Styles**.



Field descriptions for Styles

Field	Description
Sizing	Allows you to increase/decrease the height and width of the widget.
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, deselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.

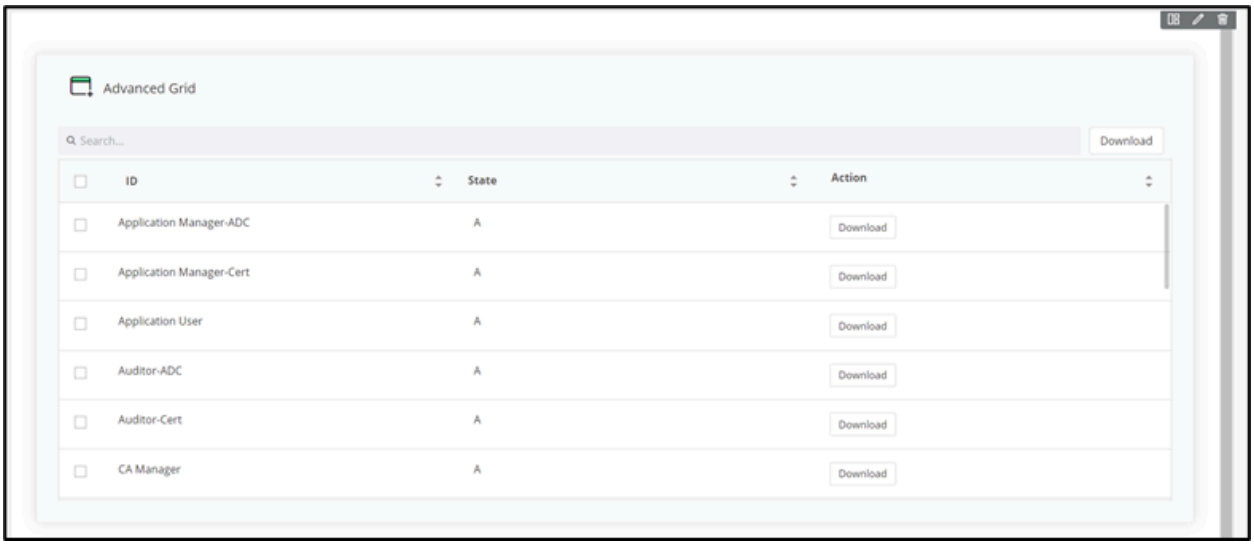
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Widget Description	The following options are available here: • **Label text Color**: Allows you to change the color of the label text. • **Label Text Size**: Allows you to select the size of the label text.

2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Grid

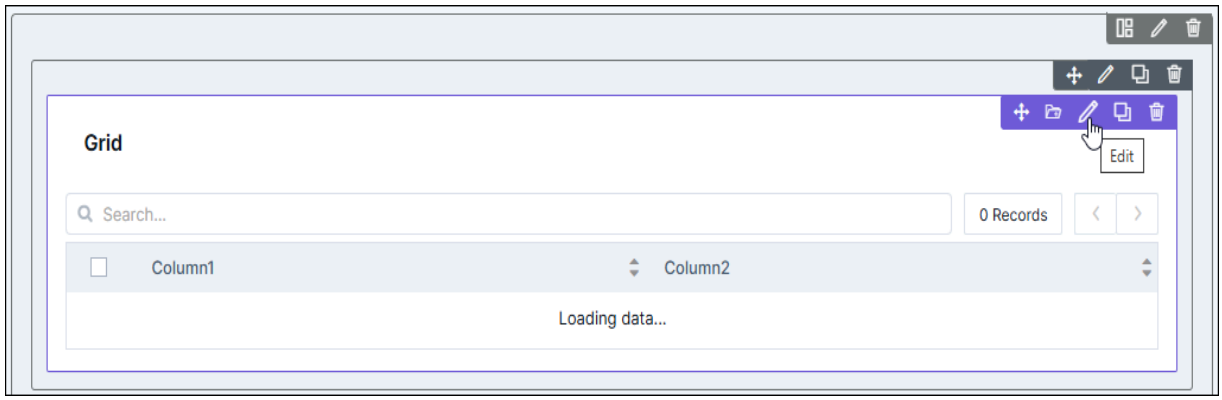
The Grid widget is an efficient and versatile tool used to display data in a tabular manner. It offers multiple actions and features that enhance data presentation and interaction. Along with common properties like widget header, icon, description, and style properties. The Grid widget includes two main features: Grid Universal Control and Column Configurations.



Configuring the Grid - Properties

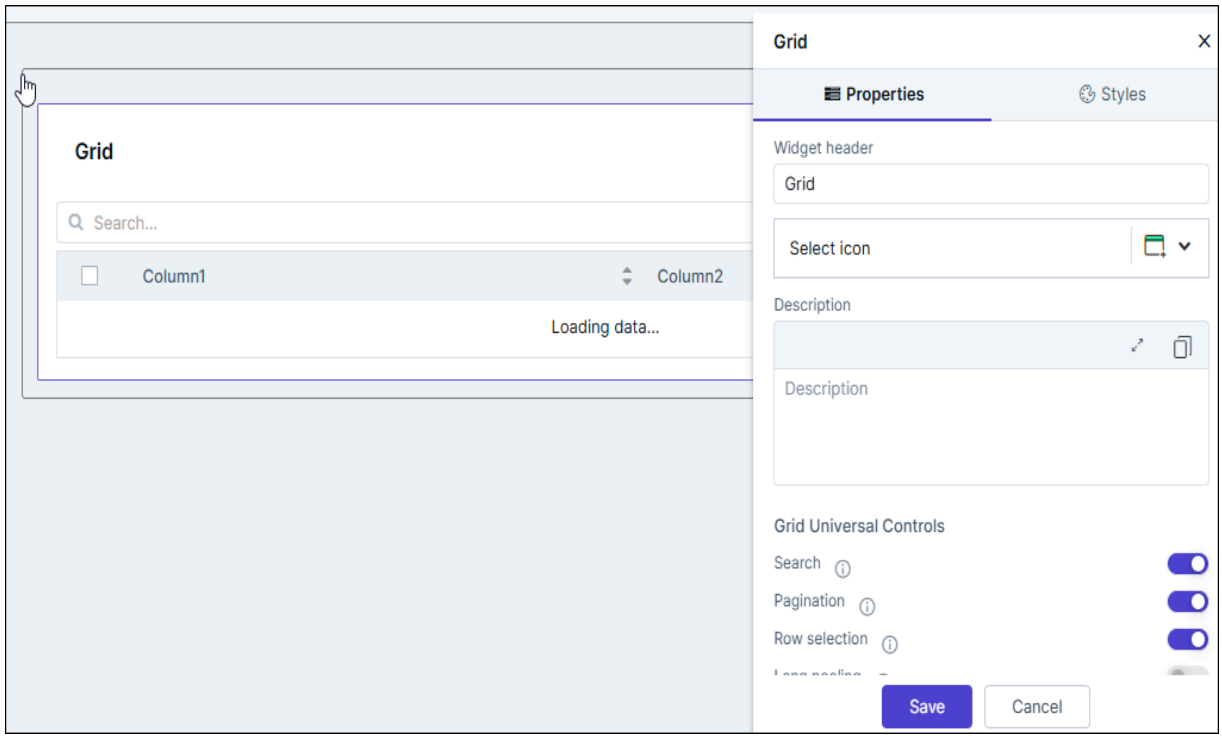
1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Grid** widget to any blue highlighted space on the page.
2. To configure the **Grid**, hover your mouse over the field and click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **Grid** pop-up window opens with the **Properties** tab open by default.

3. In the **Grid** window, under **Properties**, enter or select the required field information.



Field descriptions for Properties

Field	Description
Widget header	Allows you to rename the widget header.
Select icon	Allows you to select an icon for the widget from the options available in the dropdown.
Description	Allows you to add a description for the widget.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description

Grid Universal Controls	Allows you to enable/disable the following:
	• **Search**:Turn on the toggle for Search to display the search bar in the grid. Enabling the Search bar allows you to quickly search for a particular item in the grid. If this toggle button is enabled the **Grid Header Search Size Configuration** is displayed below the *Grid Universal Controls* section. The size of the search bar can be made
	  • **Small** (35%)
	  • **Medium** (60%)
	  • **Large** (100%) - By default, the width is set to Large.
	• **Pagination**: Turn on the toggle for Pagination to display the number of records that can be viewed on a page. Pagination enables users to navigate through large datasets efficiently, breaking the data into manageable pages.
	• **Row selection**: Turn on the toggle for Row selection to select one or more rows from the displayed data to perform specific actions.
	• **Long pooling**: Turn on the toggle for Long pooling to maintain real-time data updates. Long polling allows the server to hold the request open until new data is available. The data is retrieved from the server based on the long pooling time specified.
	  • **Long pooling time (sec)**: Enter the time in seconds for requesting data at fixed intervals.
	• **Actions**: Turn on the Actions toggle to display the **Action** buttons in the grid. The action button enables Add, Edit, and Delete operation for selected items in the grid. Refer to the section Configuring Actions in the Grid.
	• **Filters**: Turn on the Filters toggle to display the **Select Filter** dropdown next to search bar. Refer to the section Configuring Filters in the Grid.
	• **Refresh**: Turn on the Refresh toggle to display the refresh icon on the grid. Clicking this icon will refresh the data in the grid as desired.
	• **Enable Server Control**: Turn on the Enable Server Control toggle to display the "previous" and "next" icon along with the total number of displayed values. This limits the number of values displayed on the grid and clicking the "previous" and "next" icon triggers and API to server to fetch the previous or next set of records from the server.

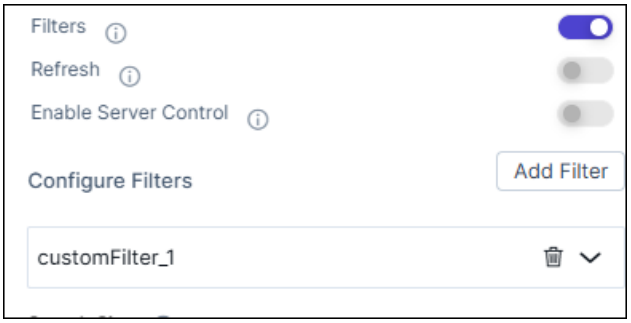
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Hooks	Allows you to assign an appropriate hook for this field. • Select a hook that is relevant to the form 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. **Note:** For more information on Adding Hooks to Form Fields, click here.
Column Preferences	Allows you to configure the columns in the grid.

4. Click **Save**.

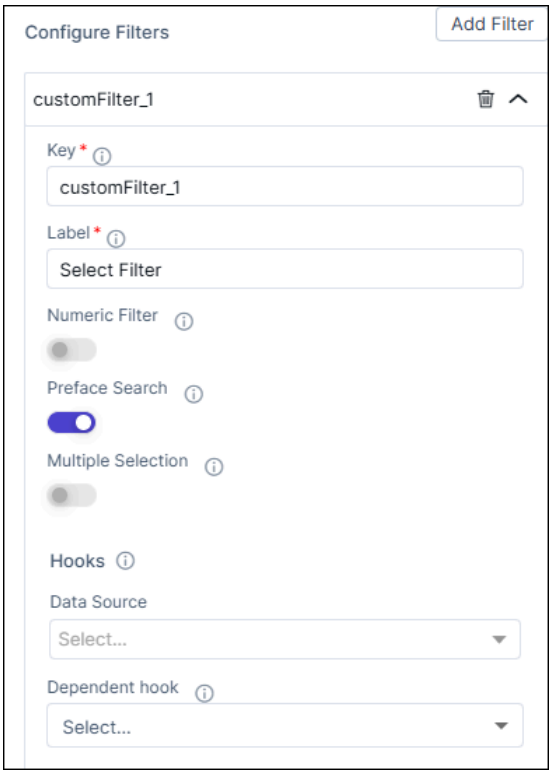
Configuring Filters in the Grid

1. If the **Filters** toggle is enabled, the **Configure Filters** section is displayed.
   The **Add Filter** button and "customFilter_1" default filter is displayed (which can be configured)



2. To add more filters, click the **Add Filter** button.
3. To configure the filters, click the (down arrow) icon.
   The following options are displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

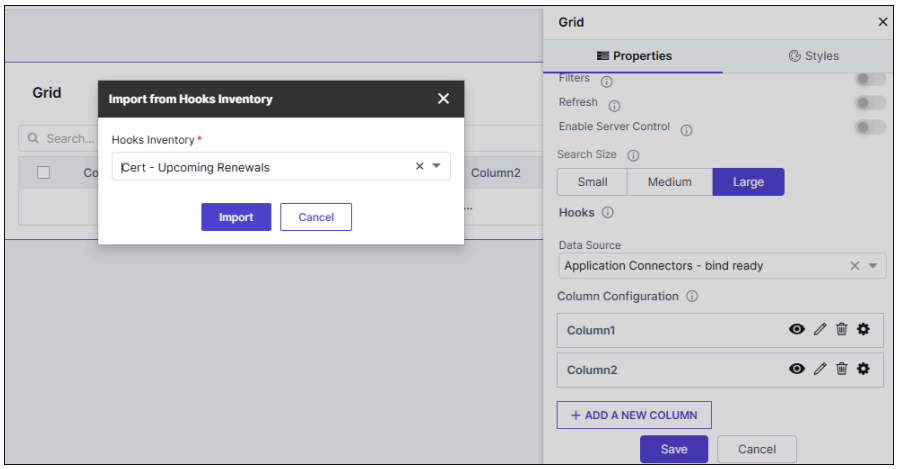


- **Key** (text field)
- **Label** (text field)
- **Numeric Filter** (toggle)
- **Preface Search** (toggle)
- **Multiple Selection** (toggle)
- **Options** (radio buttons for **Hooks**).

4. Enter the details as specified below.

Field	Description
Key	Enter a name for the filter.
Label	Enter the label that will appear in the Search Filter dropdown list.
Numeric Filter	Enable this toggle button to display numeric values (as specified in the hook) above the grid.
Preface Search	This toggle is disabled if the **Numeric Filter** is enabled. Enable this toggle to view and use the filter labels in the **Search Filter** dropdown.
Multiple Selection	This toggle is disabled if the **Numeric Filter** is enabled.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
	Enable this toggle to select multiple filter labels in the **Search Filter** dropdown.
Options	The radio button for Hook to be used for the filters. Assign a hook by selecting an option from the dropdown. For example, **Import** the **Get all roles** hook.  **Note:** For more information on Adding Hooks to Form Fields, click here.

5. Click **Save**.
   The filters are enabled and available to use in the grid.
6. To delete the filters, click (delete icon).

Configuring Actions in the Grid

The action function is used to perform header actions like download, redirect to another page, or general actions such as edit, delete, etc., for the data available in the grid.

1. If the **Actions** toggle is enabled, the **Action Configuration** section is displayed.
2. To customize colors for the Action button, the edit values in the following values:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
Action Background Color	Click (Color Palette) to chose the color scheme then click to chose the color.
Action Border Color	Click (Color Palette) to chose the color scheme then click to chose the color.
Action Text Color	Click (Color Palette) to chose the color scheme then click to chose the color.

Note: The above properties will affect all the configured Action buttons.

3. An *Action* button will be present in the Action Configuration, you may edit that using the Edit (pencil) icon or click **+Add Action** and enter the following details.

Fields	Description
Action Label	Enter a suitable label for the action to be performed.
Action Icon	Select an appropriate icon for the desired action.
Action Type	Choose from • **Download** • **Others** (default selected) • **Link**
Select Action Hook	This field is displayed only when **Action Type** = **Others** and **Download**. Click the dropdown and Import the hooks or create a new hook for the desired action.
Download Type	This field is displayed only when **Action Type** = **Download**. Select any one of file formats in which the data has to be downloaded. • **TXT (default selected)** • **YAML** • **SH** • **CSV** • **PDF**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Fields	Description
	• **ZIP** • **JSON**
Link Configuration	This field is displayed only when **Action Type** = **Link**. Select/Enter the fields as described below. • **Internal / Externa**l: Select Internal and enter the valid published internal page OR select External and enter the desired URL. • **Open link in pop-up**: Select the check box to display the link in a pop-up window. Only valid internal pages will be rendered from this option. • **Enable Pop-up Title**: Enable this toggle button to display a title for the pop-up and then type the title in the ***Enter Pop-up Title*** field. Only valid published pages will open in a pop-up. • **Choose Pop-up Position**: Choose from ***Right Aligned*** or ***Center Aligned*** • **Choose Pop-up Width**: Choose from ***Small/ Medium/ Large /Custom***. If you choose Custom then enter the ***Height*** and ***Width***.

Note: It is mandatory to have either the **Action Label** or **Action Icon** for the Action button.

4. Click **Save**.
5. To display grid actions as a dropdown, enable the **Show Grid actions as dropdown** toggle button. The default **Group** button and **+Add Group** buttons are displayed.
6. To edit the default Group, hover over the Group button and click the pencil (Edit) icon. The group configuration fields are displayed.
7. Enter details in the fields below to configure the Group button for actions.

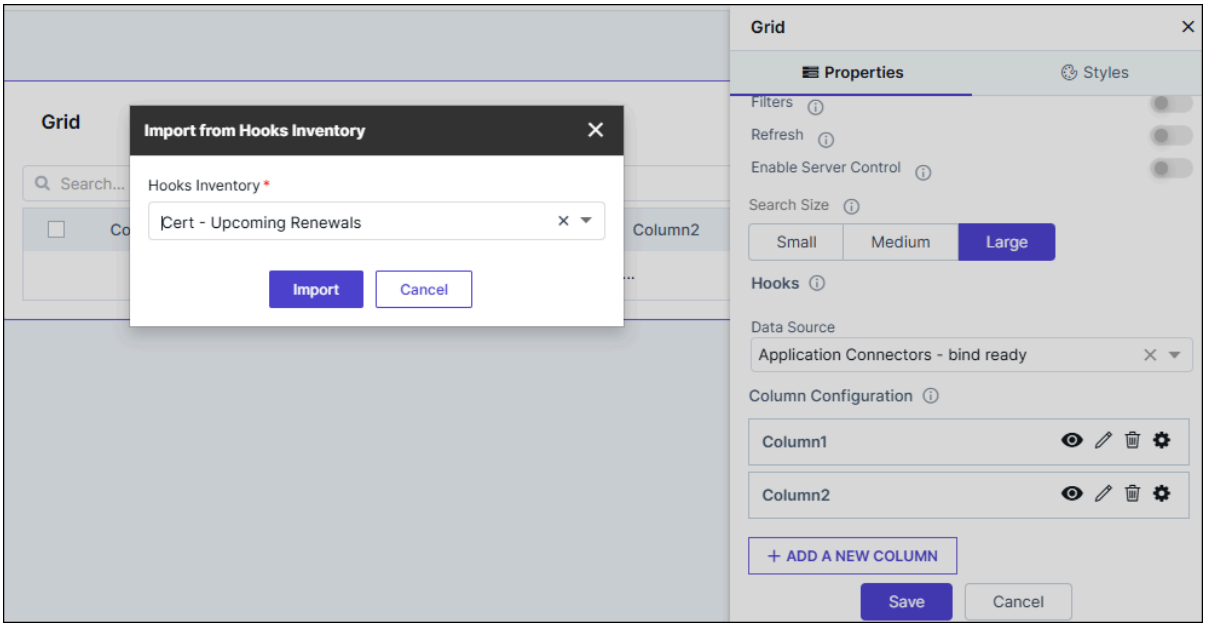
Fields	Description
Group Label	Enter a suitable label for the group button.
Group Icon	Select an appropriate icon for the group button.
Actions to be Grouped	All the Actions buttons configured will be displayed in this dropdown list. Choose as many action to add to the goup

8. Click **Save**.
9. To add action buttons to column refer to the section Configuring Columns in the Grid.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring Columns in the Grid

1. In the **Grid** window, under **Properties**, assign a hook by selecting an option from the dropdown. For example, **Import** the **Get all roles** hook.



> **Note:** For more information on Adding Hooks to Form Fields, click here.

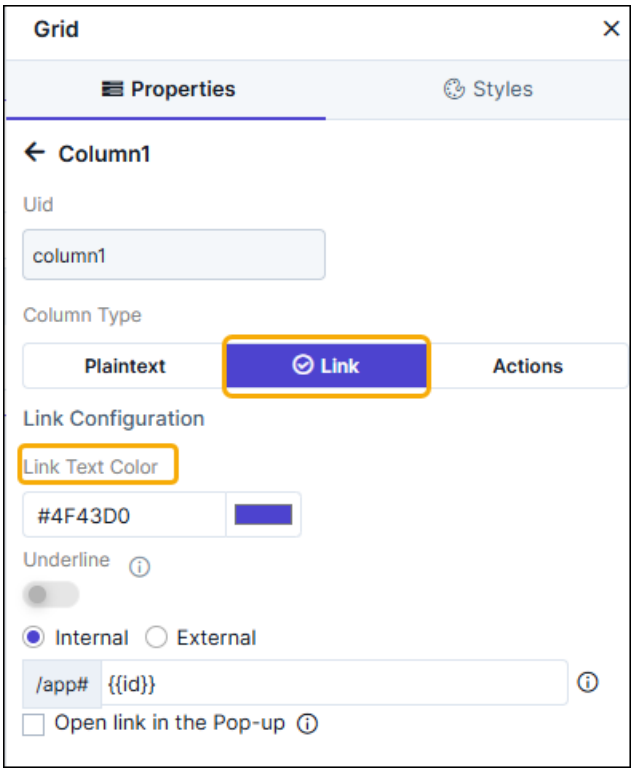
2. Under **Columns Preferences**, to change the name of the first column, click **(Edit)** icon.
3. Type the name of the column as required and click .
4. To configure the **Column Type**, click icon.

   The following options are available:

   - **Plaintext**: Select this option to display the column items as plain text.
   - **Link**: Select this option to add links to each column item.
   - **Actions**: Select this option to add actions such as Download or Delete to the columns.
5. To configure links for the column items, under **Column Type**, click **Link**.

   The **Link Configuration** option is displayed.

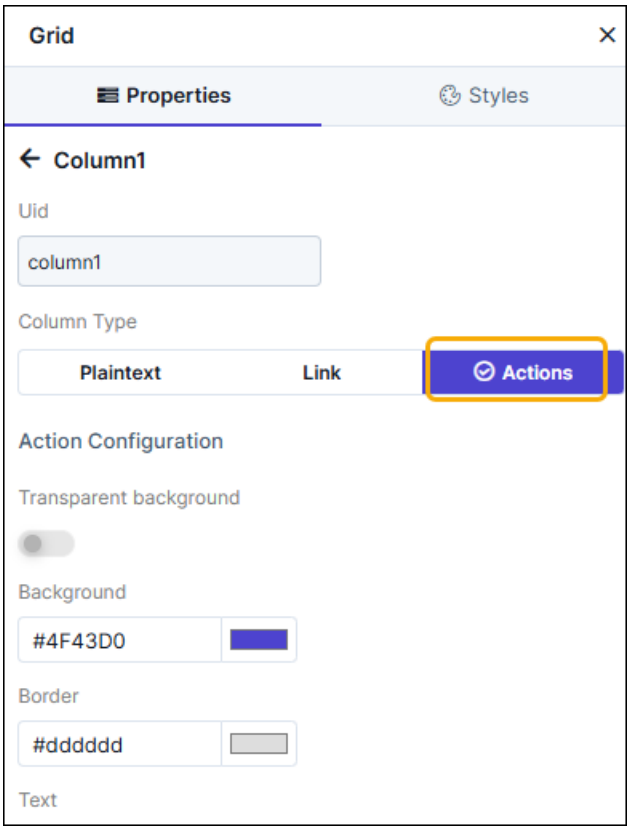
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. Select the color for the **Link Text**.
7. Select the **Text Underline** checkbox to underline the link text.
8. Select the type of link to be assigned to the column fields. For example, **Internal**.
9. Clicking on any role under the **Name** column, will redirect to the linked page.
10. To configure an action for a column, under **Column Type**, click **Actions**.

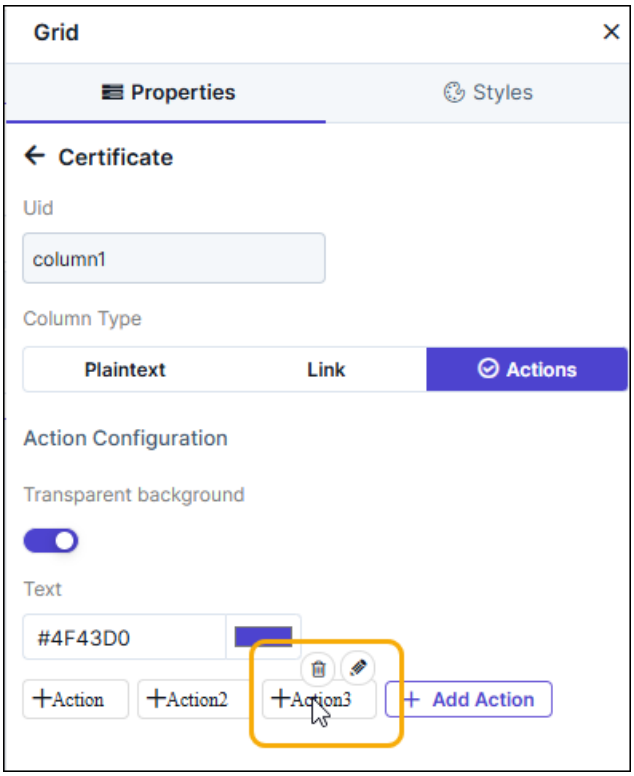
    The **Action Configuration** option is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



11. Select the Action **Background** Color.
12. Select the Action **Border** Color.
13. Select the Action **Text** Color.
14. To add an action for the column field, hover your mouse over **Action** and click **(Edit)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



More fields to add the action are displayed as shown.

15. Enter the **Action label**. For example: **Download**.
16. Select the **Action Type** as **Download** or **Others**. For example, select Download to download the list of roles to your machine as a .csv file.
17. Select the hook to be assigned to the action. For example: **Role Download**.

> **Note:** For more information on Adding Hooks to Form Fields, click here.

18. Click **Save**.
    The configured action is added to the grid.

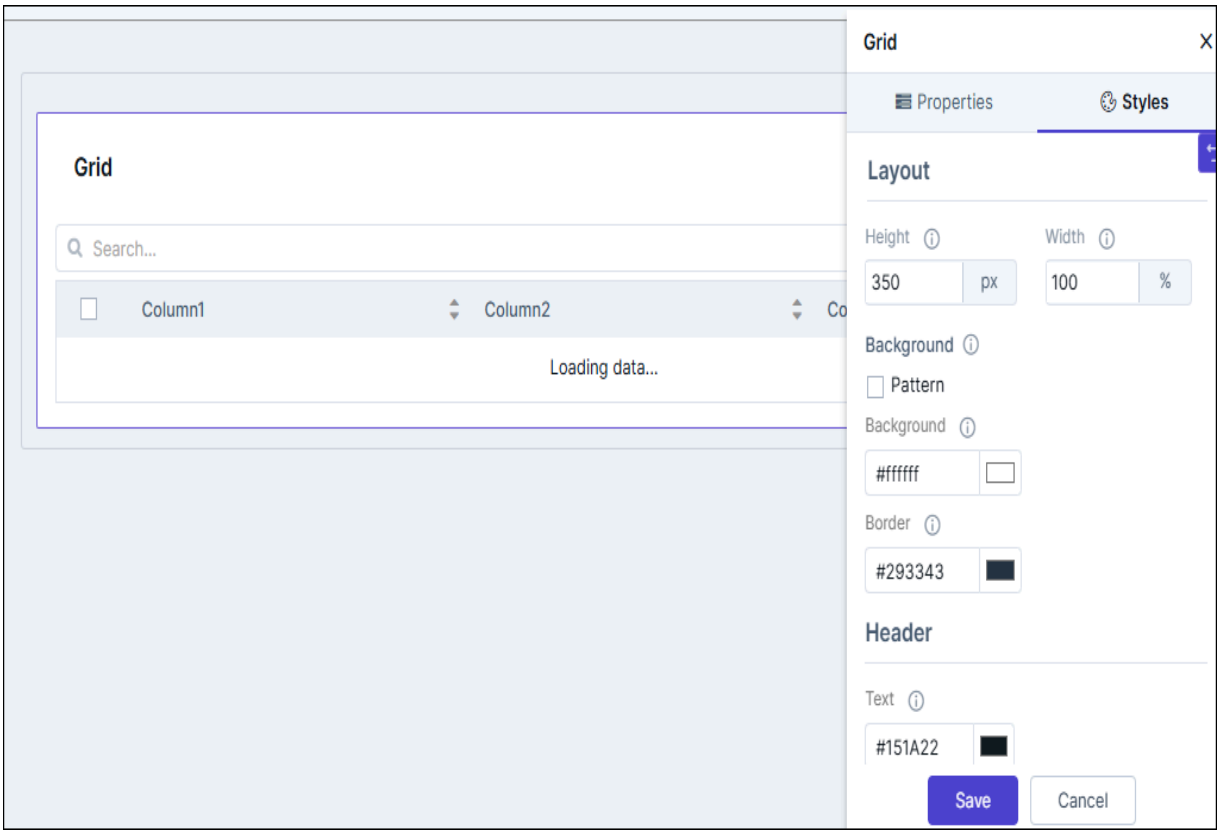
> **Note:** You can also add actions to the grid by turning on the **Actions** toggle under **Grid Universal Controls**.

19. To add more actions to the grid, under **Action Configuration**, click **Add Action**.
20. For options on configuring this action, hover your mouse over the action and click **(Edit)** icon.
21. To delete this action, hover your mouse over the action and click **(Delete)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Grid - Styles

1. For more options to customize the Grid component, in the **Grid** window, click **Styles**.



Field descriptions for Styles

Field	Description
Sizing	Allows you to increase/decrease the height and width of the widget.
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, deselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

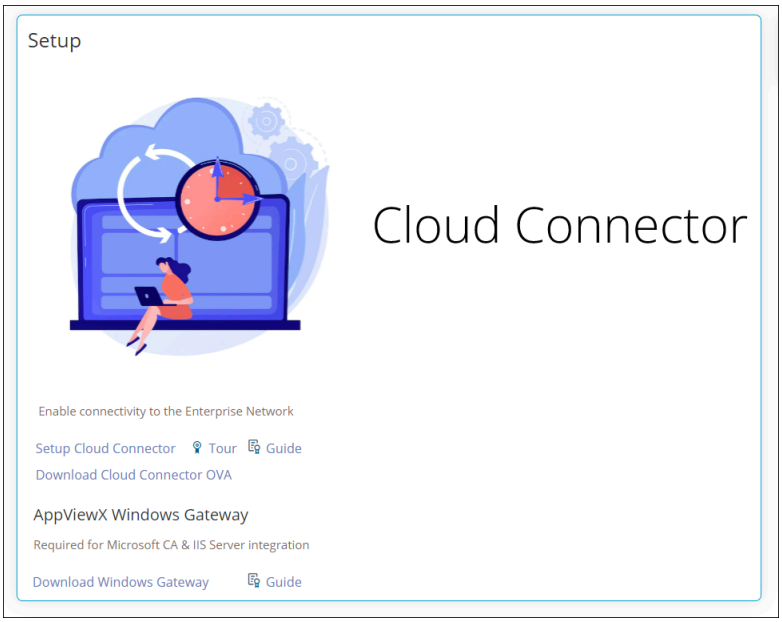
Field	Description
Tabular Header Style	The following options are available here: • **Header Background Color** - Select the background color for the header from the available options. • **Header Text Color** - Select the header text color from the available options. • **Header Text Size** - Select the header text size from the available options. • **Header Text Alignment** - Select the header alignment as left, right or centre.
Tabular Row Style	The following options are available here: • **Row Background Color** - Select the background color for the row from the available options. • **Row Color** - Select the row color from the available options. • **Row Text Size** - Select the row text size from the available options. • **Row Text Alignment** - Select the row alignment as left, right or centre.
Icon Style	The following options are available here: • **Icon Color** - Select the Icon Color from the available options. • **Icon Background** - Select the Icon Background from the available options. • **Icon Background Color** - Select the Icon Background Color from the available options.

2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Card

The Card component is a versatile container that allows users to group multiple smaller components, including lists, buttons, rich text editors, and images within a single cohesive element. It offers comprehensive customization options to give users maximum control over the look and feel of the components it contains. The card component also features an accordion functionality for further organization and interaction.

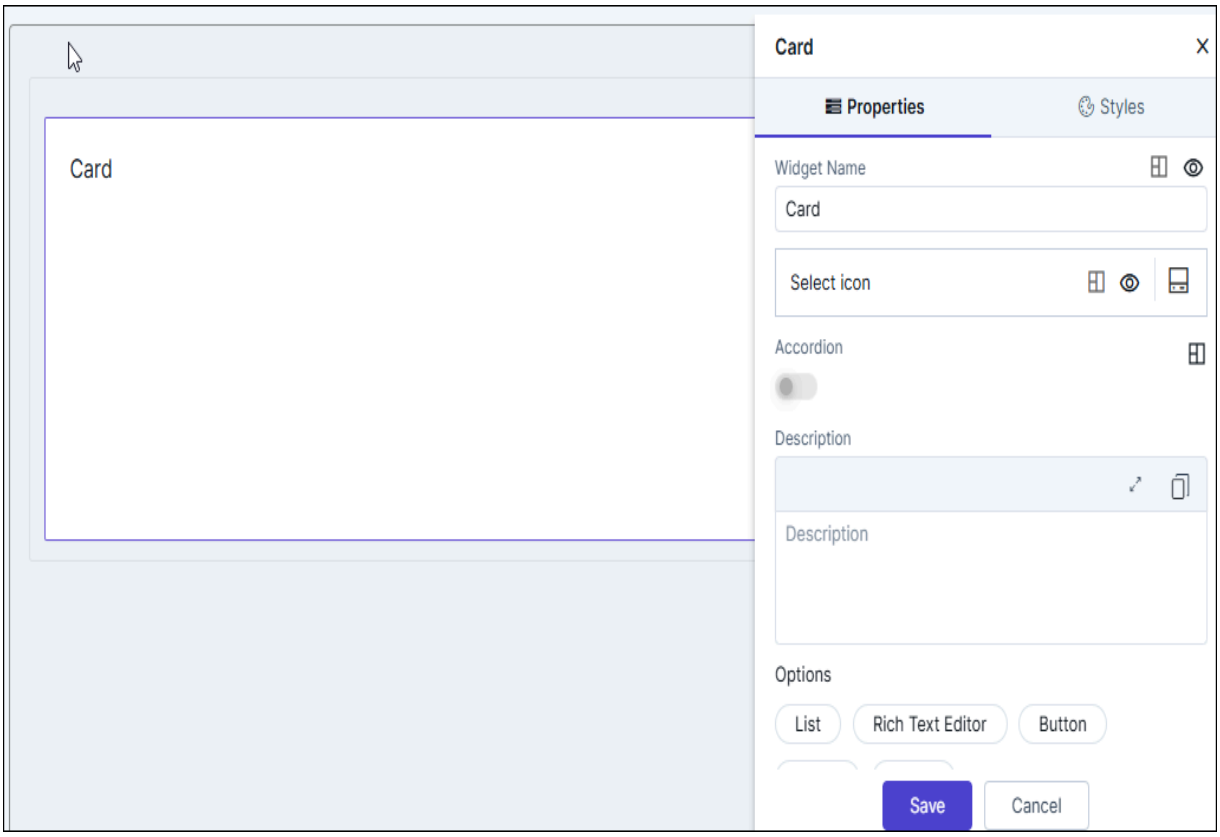
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Configuring the Card - Properties

1. From the **Build** component in the left menu, under the **Widgets** section, drag and drop the **Card** widget to any blue highlighted space on the page.
2. To configure the **Card** widget, hover your mouse over the widget and click **(Edit)** icon.
   The **Card** pop-up window opens with the **Properties** tab open by default.
3. In the **Card** widget window, under **Properties**, enter or select the required field information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Field descriptions for Properties

Field	Description
Widget Name	Allows you to enter the name for the card widget. Styling options available for the Widget name are: • - Allows you to align the widget name to the left, right, center of the widget. • / - Allows you to show or hide the widget name.

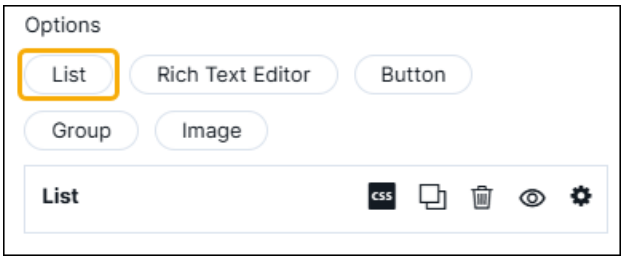
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Select icon	Allows you to select an icon for the widget. Styling options available for the icon are: • - Allows you to align the icon to the left, right, or adjacent to the widget name. • / - Allows you to show or hide the icon.
Accordion	• Turn this toggle on or off to hide or show large volumes of content in the card widget. • Turning the toggle on displays options to select icons to Expand and Collapse the content in the card widget. • Click to align the Expand and Collapse icons to the left or right of the widget.
Description	Allows you to add a description for the card widget.
Options	Allows you to add the following components to the card widget: • **List** • **Rich Text Editor** • **Button** • **Group** • **Image**.

4. Click **Save**.

Configuring Lists in the Card Widget

1. In the **Card** window, in the **Properties** tab, under **Options**, click **List**.
   The **List** row is displayed.

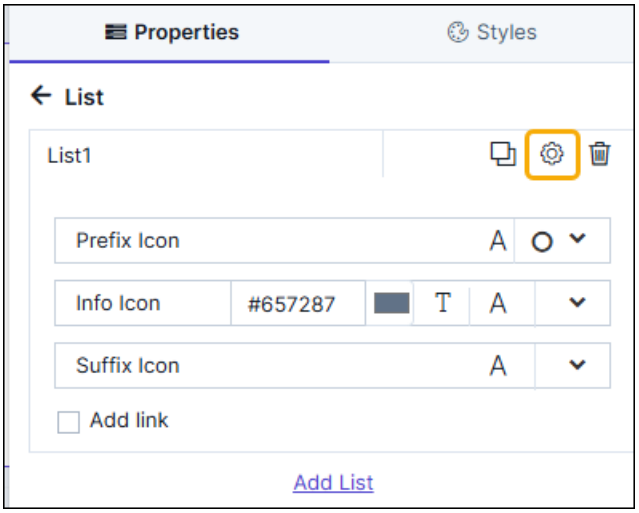


2. To expand the row and add a list, click **(Add)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A new row is displayed with the default name as **List1**. You can rename this by typing over the existing name.

3. In the **List1** row, to configure the list, click **(Settings)** icon, and again click the settings icon within the list configuration.
   The following options are displayed as shown in the screenshot below.
   - **Prefix Icon**
   - **Info Icon**
   - **Suffix Icon**
   - **Add link**.



Options descriptions

Option	Description
Prefix Icon	• To add a prefix icon next to the list name, click and select the required icon from the options available. • To increase or decrease the size of the prefix icon, click A and set the icon size.
Info Icon	• To add an Info icon next to the list name, click and select the required icon from the options available. • To increase or decrease the size of the Info icon, click A and set the icon size.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
	• To select a color for the info icon, click the color picker box and select the required color. • To add info text, click and enter the text in the textbox.
Suffix Icon	• To add a suffix icon next to the list name, click and select the required icon from the options available. • To increase or decrease the size of the suffix icon, click and set the icon size.

4. To add a link for the list, select the **Add link** checkbox.
   Option to add **Internal**, **Others** (External), and **Workflow** link (s) is displayed.

> **Note:** For more information on adding **Internal** link(s), click here.

> **Note:** For more information on adding **Others** (External) link(s), click here.

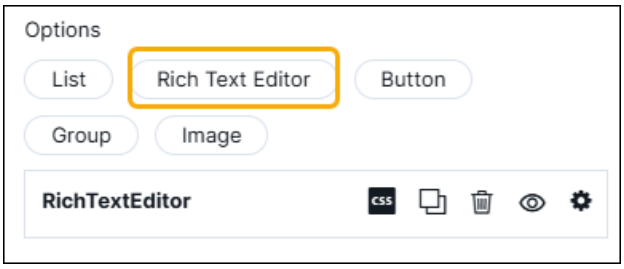
> **Note:** For more information on adding **Workflow** link(s), click here.

5. To hide the List row, click **(View)** icon. To display it again click **(Hide)** icon.
6. To delete the List row, click **(Delete)** icon.
7. To define the padding, margins, background color, and font size for the List within the card, click **(Layout)** icon.
8. Once you have configured all the settings, click **Save**.
9. To add another list, click **Add List**.

Configuring Rich Text Editor in the Card Widget

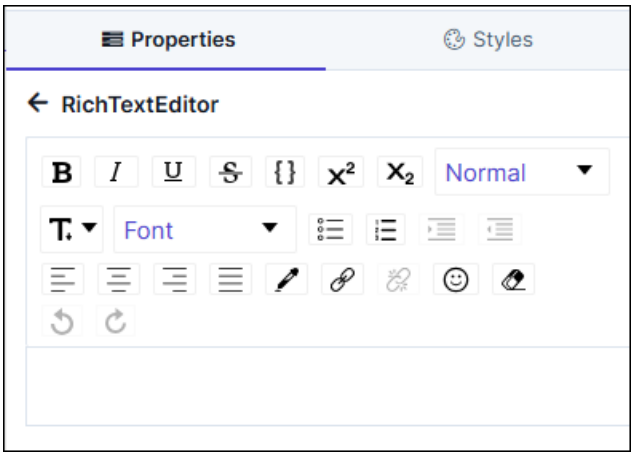
1. In the **Card** window, in the **Properties** tab, under **Options**, click **Rich Text Editor**.
   The **List** row is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. To expand the row and add a Rich Text Editor, click **(Settings)** icon

   The **Rich Text Editor** is displayed.



> **Note:** For more information on the Rich Text Editor, click here.

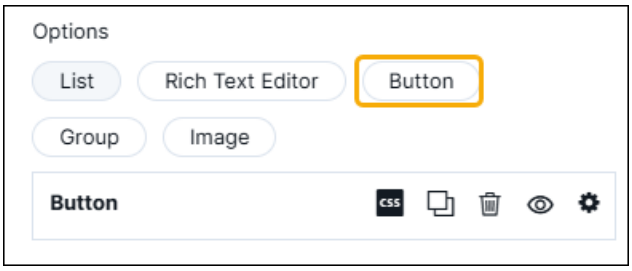
3. To hide the Rich Text Editor, click **(View)** icon. To display it again click **(Hide)** icon.
4. To delete the Rich Text Editor, click **(Delete)** icon.
5. To define the padding, margins, background color, and font size for the Rich Text Editor within the card, click **(Layout)** icon.
6. Once you have configured all the settings, click **Save**.

Configuring a Button in the Card Widget

1. In the **Card** window, in the **Properties** tab, under **Options**, click **Button**.

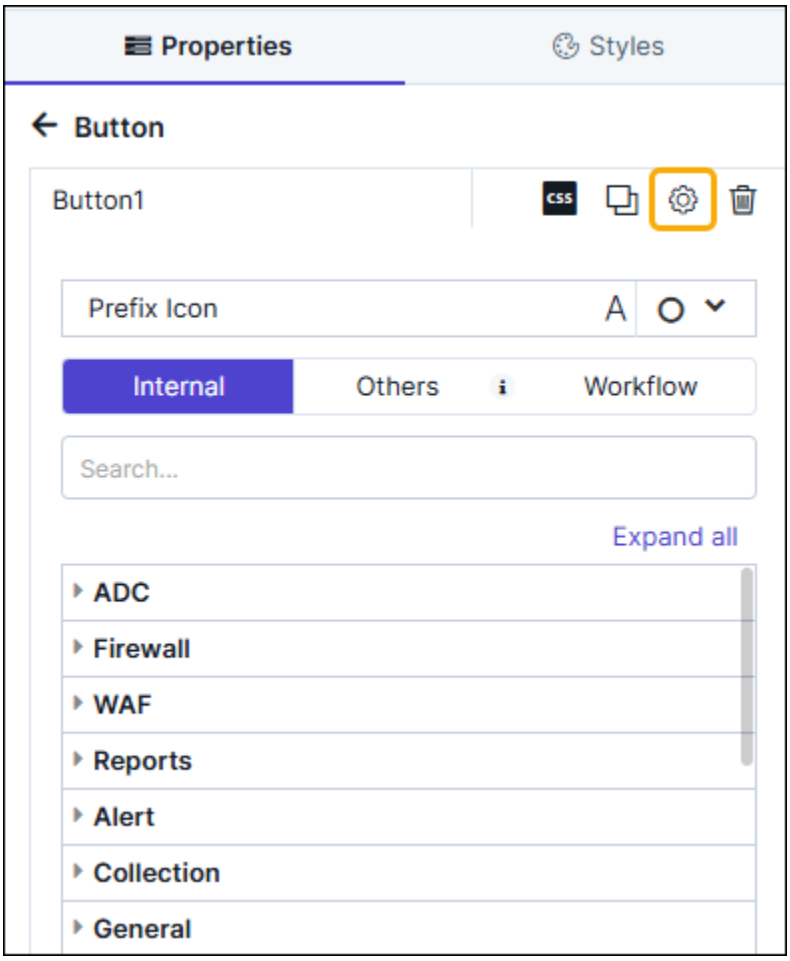
   The **Button** row is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. To expand the row and add a list, click **(Settings)** icon.
3. In the **Button1** configuration, to configure the list, click **(Settings)** icon.

   The following options are displayed as shown in the screenshot below:



- **Prefix Icon** - To add a prefix icon next to the Button click and select an icon from the options available. To change the size of the icon, click .
- **Add link** - Allows you to add **Internal**, **Others** (External), and **Workflow** link(s) for the Button.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note:

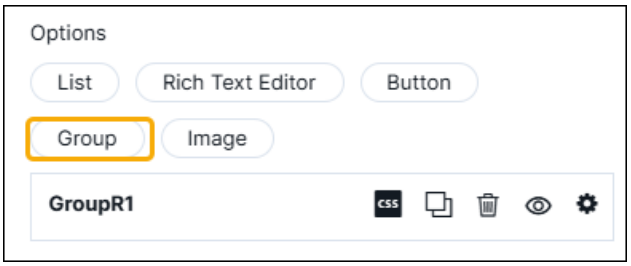
- For more information on adding **Internal** link(s) to a Button, click here.
- For more information on adding **Others** (External) link(s) to a Button, click here.
- For more information on adding **Workflow** link(s) to a Button, click here.

4. To hide the Button row, click **(View)** icon. To display it again click **(Hide)** icon.
5. To delete the Button row, click **(Delete)** icon.
6. To define the padding, margins, background color, font size, and alignment for the Button within the card, click **(Layout)** icon.
7. Once you have configured all the settings, click **Save**.
8. To add another button click **Add Button**.

Configuring Grouping in the Card Widget

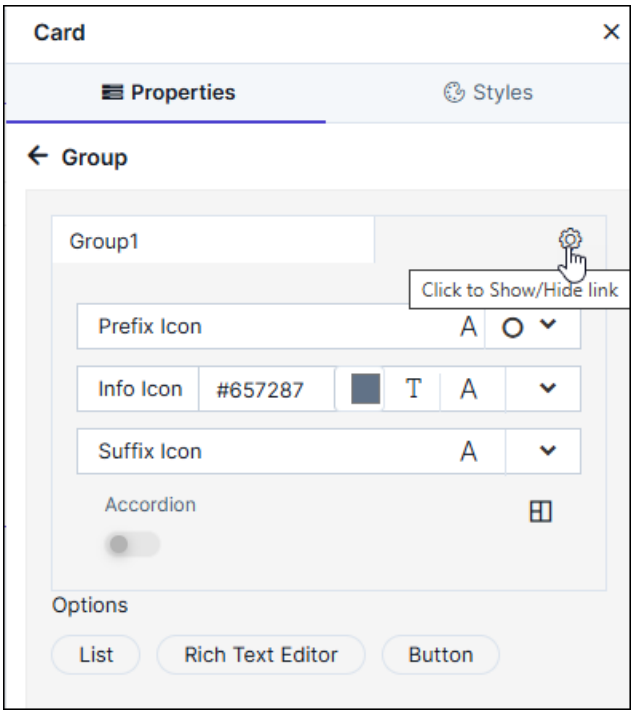
This feature allows you to group card elements such as list(s), rich text editor, and button(s) within a group.

1. In the **Card** window, in the **Properties** tab, under **Options**, click **Group**.
   The **Group** row is displayed. You can rename this by typing over the existing name.



2. To expand the row and add a list, click **(Settings)** icon.
3. In the **Group** configuration, to configure the group, click **(Settings)** icon.
   The following options are displayed as shown in the screenshot below:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



- **Prefix Icon**
- **Info Icon**
- **Suffix Icon**
- **Accordion**.

Options Descriptions

Option	Description
Prefix Icon	• To add a prefix icon next to the list name, click and select the required icon from the options available. • To increase or decrease the size of the prefix icon, click A and set the icon size.
Info Icon	• To add an Info icon next to the list name, click and select the required icon from the options available. • To increase or decrease the size of the Info icon, click A and set the icon size.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
	• To select a color for the info icon, click the color picker box and select the required color. • To add info text, click T and enter the text in the textbox.
Suffix Icon	• To add a suffix icon next to the list name, click ⌄ and select the required icon from the options available. • To increase or decrease the size of the suffix icon, click A and set the icon size.
Accordion	Allows you to hide or show large volumes of content in the card widget by turning this toggle on or off. You can select icons to Expand and Collapse the content in the card widget and also set the placement of these icons.

4. To add a **List**, **Rich Text Editor**, or **Button** in the Group, click on the respective link under **Options**.

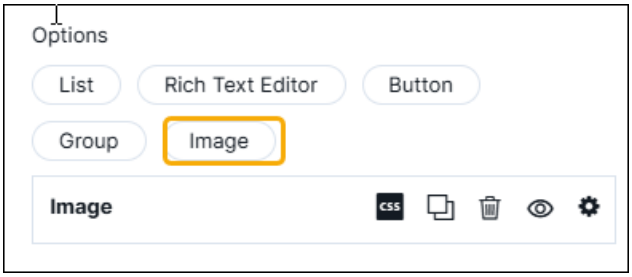
> **Note:** For steps on configuring a list, click here.

> **Note:** For steps on configuring a Rich Text Editor, click here.

> **Note:** For steps on configuring a Button, click here.

Configuring Images in the Card Widget

1. In the **Card** window, in the **Properties** tab, under **Options**, click **Image**.
   The **Image** row is displayed.



2. To configure an image, click ⚙ **(Settings)** icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

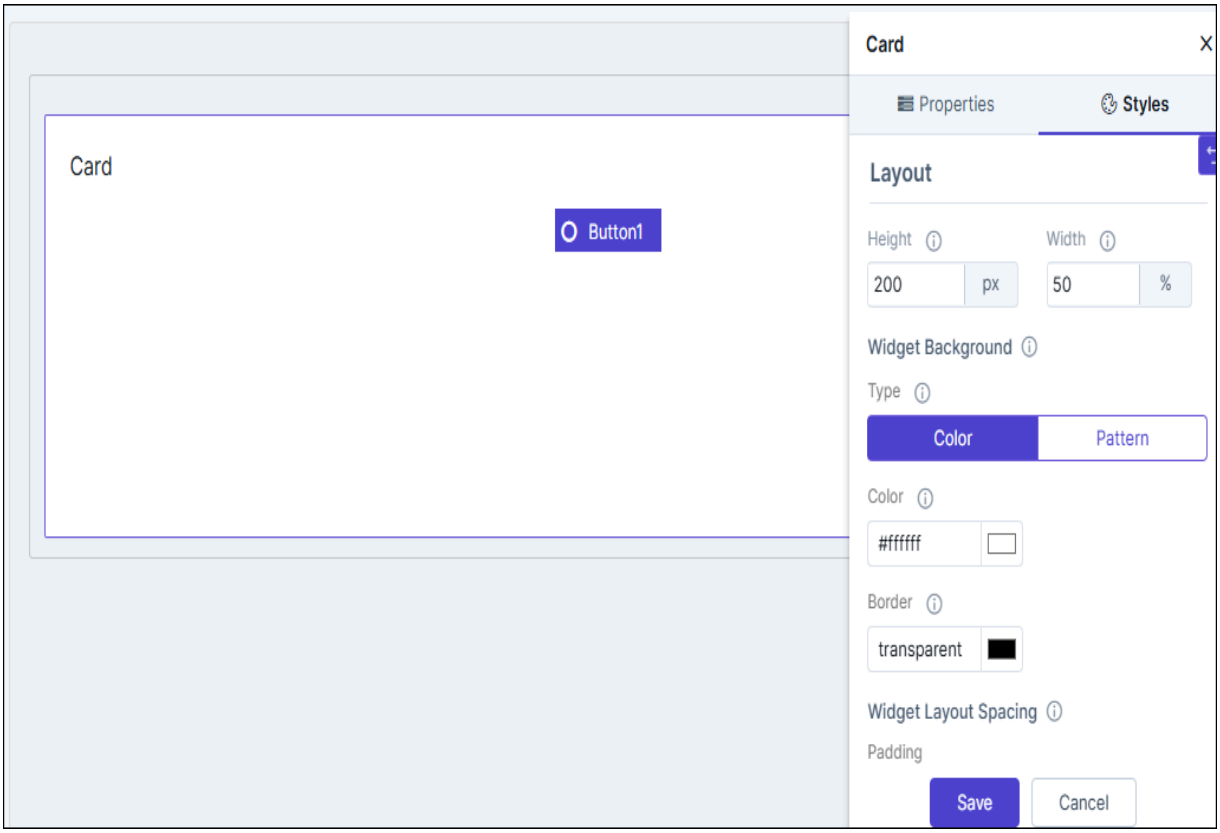
3. To upload an image in the card widget, click **(Upload)** icon and upload the required image from your computer.

> **Note:** Allowed image formats are: .svg, .jpeg, and .png

4. To delete this image and select another, click **(Delete)** icon.
5. To add another image, under **Options**, click **Image**.
   A new **Image** row is displayed.
6. Once you have added the required image(s), click **Save**.

Configuring the Card - Styles

1. For more options to customize the Card component, in the **Card** window, click **Styles**.



Field descriptions for Styles

Option	Description
Sizing	Allows you to increase/decrease the height and width of the widget.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Option	Description
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, deselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.
Widget Description	The following options are available here: • **Label text Color**: Allows you to change the color of the label text. • **Label Text Size**: Allows you to select the size of the label text.
Icon Style	Allows you define the following style settings: • **Icon Color** - You can select the color for the icon from the available options. • **Icon Background** - You can select the background for the icon from the available options. • **Icon Background Color** - You can select the background color for the icon from the available options.
Widget Layout Style	Allows you to define the padding and margin of the Card widget.

2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Form

The Form widget is a dynamic and interactive component that allows users to gather input from users and execute specific actions upon form submission or other form-related events. This widget acts as a container, enabling users to add multiple components within it to collect data and facilitate various operations.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Key Features of the Form Widget:

1. **Input Collection:** The Form widget supports various components, such as text fields, checkboxes, radio buttons, dropdown menus, and more, to gather user inputs. Users can customize these components to suit their specific data requirements.
2. **Form Actions:** Upon form submission or other form-related events, users can define actions to be executed. These actions may include data processing, data storage, sending notifications, or triggering additional workflows.
3. **Validation:** The Form widget may incorporate built-in or custom validation rules to ensure data integrity and accuracy. Users can specify mandatory fields, data formats, and other validation criteria.
4. **Dynamic Rendering:** Depending on user interactions or selections, the Form widget can dynamically render or update certain components, providing a responsive and adaptive user experience.
5. **Error Handling:** In case of validation errors or other issues, the Form widget can display relevant error messages to guide users in correcting their inputs.
6. **Styling and Layout:** Users can apply styling options to the Form widget, such as background colors, borders, spacing, and dimensions, to ensure consistency with the overall design of the application.

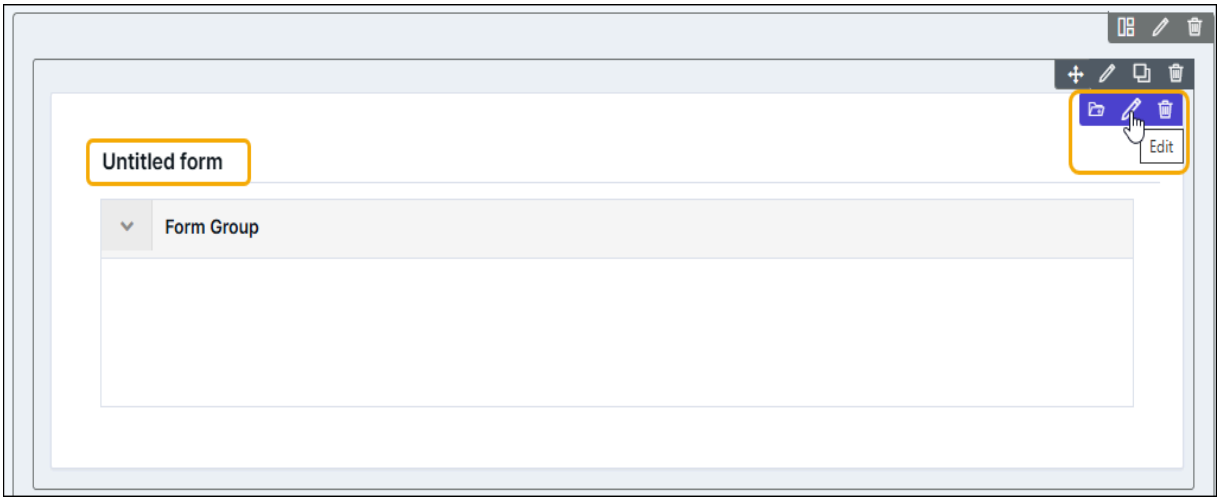
- Configuring a Form
- Form Field - Label
- Form Field - Text Box
- Form Field - Password
- Form Field - Date Time Picker
- Form Field - Text Area
- Form Field - Radio Button
- Form Field - Checkbox
- Form Field - Select Box
- Form Field - Multi-select Box
- Form Field - Form Button
- Form Field - Download
- Form Field - File Upload
- Form Field - Script Editor
- Form Field - Rich Text Editor
- Form Field - Tabular
- Form Field - Form Group

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring a Form

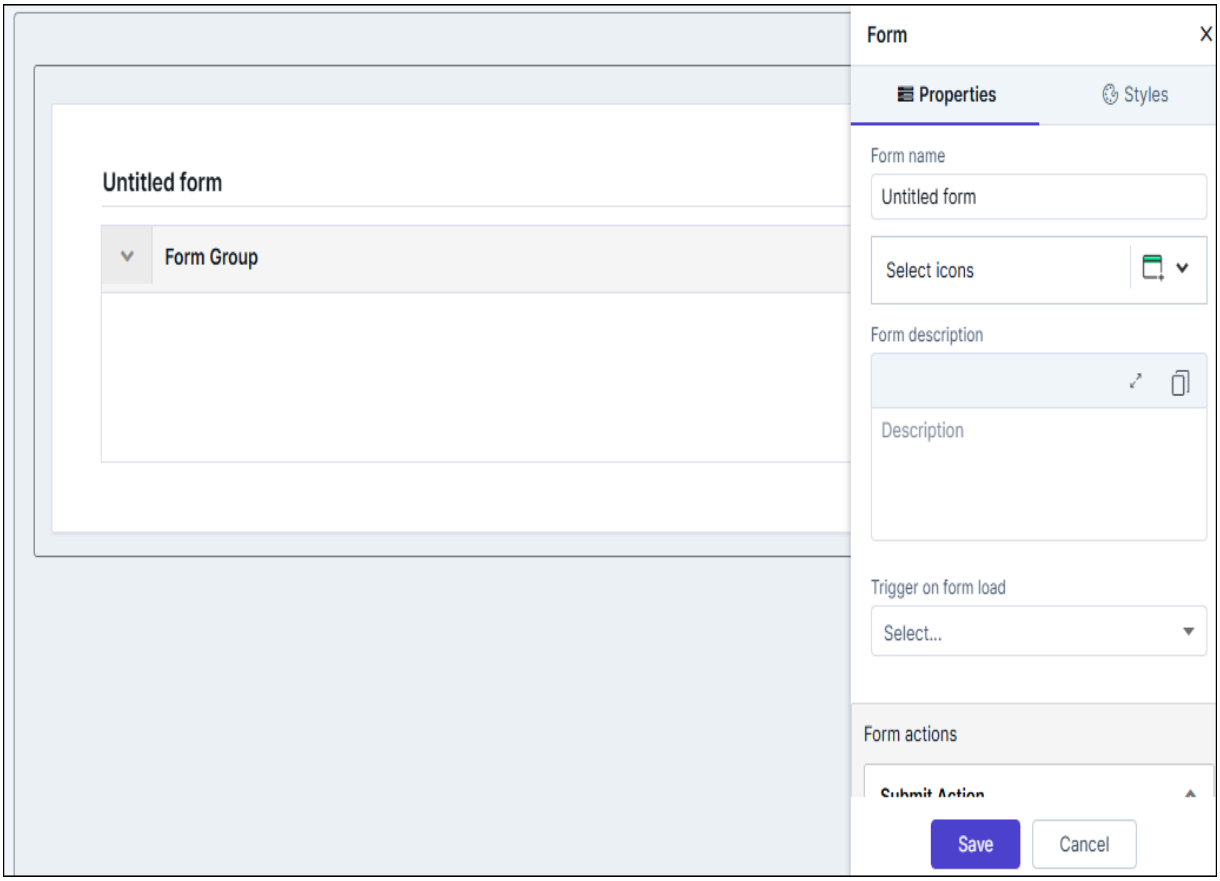
To set the Form properties,

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select any of the form components and drag it to the right on the empty new page.
   The **Untitled form** page is created with icons to *Save*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the form properties and styles click the Edit icon. Refer to the sections below.



- Assigning Hooks to Form Fields

Editing Form Properties

The form's edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Form name	Enter a suitable name for the form. The name is displayed on the top left of the form.
Select icons	Click the dropdown to choose the icons to display before the form name.
Form description	Enter a suitable description; displayed below the form name.
Select scripts to run on form load	The hooks associated with all the added form components are displayed in the dropdown.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Reset Action	To display a Reset button on the form turn ON the toggle button. This button reset all the form fields (components) to blank or default value.
Button label name	This field is displayed only if the **Reset Action** is turned ON. The default value is "Reset" to change the value, enter the desired name.
Action Alignment	Choose from **Left**, **Center**, or **Right** to display the action buttons (Submit, Reset, etc) as per the required alignment.
Form Layout	Choose the following: • **Horizontal** - the component labels and fields to be displayed in the same line. • **Vertical** - the component label is displayed above the field.
Group Options	The properties of this component are enabled when the Form Group component is added to the form. Refer Form Group for more details
Form Actions (Submit)	
Button label name	Enter a name for the button. The default value is "Submit".
Data Source	Hooks associated with the component will be displayed in the dropdown. Select any. To import hooks from the hooks inventory click **Import** or to create a new hook, click **Create New**.
Enable page redirect	Turn ON the toggle button to perform a redirect operation on the button.
Redirect URL	This field is displayed only if the **Enable page redirect** is turned ON. Enter a URL for redirect on click of Submit button. **Note:** If the **Data Source** (hooks) and **Redirect URL** are both enabled for the form. The Data Source takes the highest priority.
Open link in the pop-up	This checkbox is displayed only if the **Enable page redirect** is turned ON. To view the redirected URL on a separate window, click the checkbox..
Enable Pop-up Title	To display a title for the window, turn ON the toggle button.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Enter Pop-up Title	This field is enabled when the Enable Pop-up Title is turned ON. Enter a suitable title for the pop-up window.
Choose Pop-up Position	This field is enabled when the Enable Pop-up Title is turned ON. Choose **Right Aligned** or **Center Aligned** for the pop-up window.
Choose Pop-up Width	This field is enabled when the Enable Pop-up Title is turned ON. Choose **Small**, **Medium**, or **Large** size for the pop-up window.
Disable action	Allows to disable the Submit button. (If the button is disabled, it can be configured in the final published form to become enabled either after all the required details are entered or based on the conditions defined in the hooks.).
Button Theme	This option is enabled only when new action buttons are added. The **primary** button, displayed in violet, represents the main action on the form, while the **secondary** buttons, shown in white, are used for less critical actions. **Note:** The Submit action is default Primary on the form.

Editing Form Styles

The form's edit panel opens on the right, click the **Styles** tab to configure the overall form appearance.

Field description for Styles

Fields	Description
Theme	Choose the following • **Theme 1** - the form components have a thinner border on mouse rollover. • **Theme 2** - the form components have a thicker border on mouse rollover.
Widget Background	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Styles (continued)

Fields	Description
Type	This setting is used to change the background color for the entire form. • If you choose as **Color**, then in the sub fields **Color**, pick the desired color. • If you choose **Pattern**, then pick the desired pattern from the options shown.
Border	This setting is used to change the outer border color for the entire form. Pick a color from the color-picker.
Header	
Color	Choose the text color for the Form Name.
Size	Choose a text size for the Form Name.
Description - Color	Choose a text color for the Form Name's Description
Description - Text Weight	Choose a text weight (Normal, Semi-Bold, Bold, Black) for the Form Name's Description
Border Bottom - Border Color	Choose a color for the border below the Form Name.
Border Bottom - Border Width	Choose a width (numeric) for the border below the Form Name.
Icon Style	Choose from the four colors for the icon displayed before the form name.
Form Group*(To maintain a common style for the Form Group components added in the form, the style settings for the Form Group lies within the Form).*	
Header - Color	Choose the text color for the Form Group's name.
Header - Size	Choose the text size for the Form Group's name.
Header - Weight	Choose the text weight (Normal, Semi-Bold, Bold, Black) for the Form Group's name.
Border & Spacing - Color	Choose a color for the border below the Form Group component.
Border & Spacing - Width	Choose a color for the border width below the Form Group component.
Header Description - Color	Choose a text color for the Form Group's description.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Styles (continued)

Fields	Description
Header Description - Size	Choose a text size for the Form Group's description.
Header Description - Weight	Choose a text weight (Normal, Semi-Bold, Bold, Black) for the Form Group's description.
Content *(To maintain a common style for the Form components added in the form, the style settings pertaining to the component names and fields lies within the Form).*	
Form Field Label - Color	Choose a text color for the labels of the form components.
Form Field Label - Size	Choose a text size for the labels of the form components.
Form Field Label - Weight	Choose a text weight (Normal, Semi-Bold, Bold, Black) for the labels of the form components.
Form Field - Color	Choose a color for text that is entered with the form fields (text box, description box etc).
Form Field - Size	Choose a size for text that is entered within the form fields (e.g.: text box, description box etc).
Form Field - Weight	Choose a weight (Normal, Semi-Bold, Bold, Black) for text that is entered with the form fields (e.g.: text box, description box etc).
Form Field - Border Color	Choose a color for the form component's border.
Form Field - Field Height	Choose the height for the form component's border.
Form Field - Field Border Radius	Choose the border radius (curved borders) for the form component's border.
Field Size	Choose the overall field size (Small, Medium, Large) for the form components. • Small - size less than ½ of the form. • Medium - size about ¾ of the form. • Large - size of the form's length.

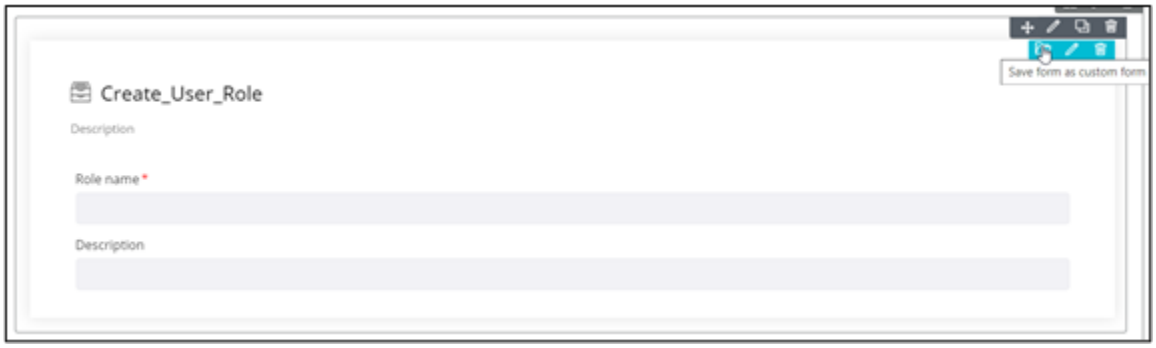
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Saving Forms to Custom Folder

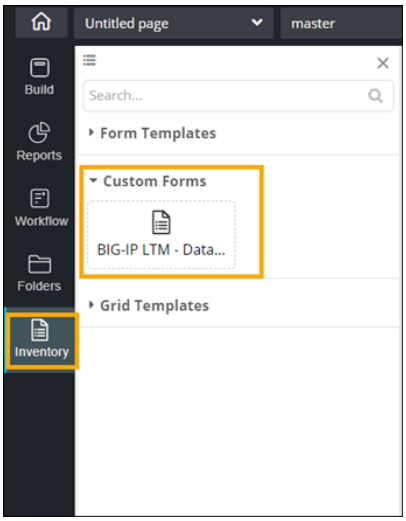
Any form created from scratch or a customized form template can be saved to the Custom folder to be reused later.

To save a form in the custom folder:

1. Hover your mouse over the form and from the top right corner, click **(Folder)** icon.



2. In the **Confirmation** pop-up window, click **Yes**.
   The form is added to the **Custom** folder in the **Inventory** section.



3. To reuse a custom form, simply drag and drop the required form from the **Custom** folder into the canvas.

Assigning Hooks to Form Fields

You can assign hooks to the form fields or to the form action buttons (eg. Submit) 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.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

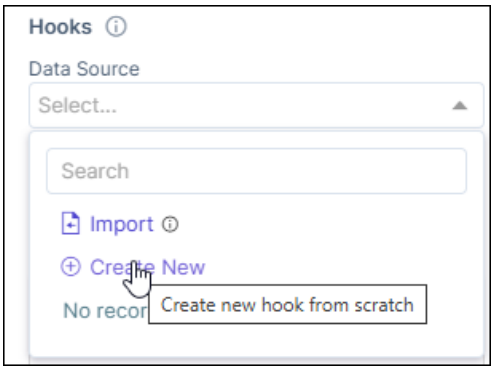
1. To assign a hook to a form field, hover over the form field and click **(Edit)** icon.
2. In the field's pop-up window, under **Properties**, scroll down to **Hooks**.

   The following options are available under Hooks:

   - **Select an existing hook:** You can select an existing hook from the available options and assign it to the form field to fetch data.
   - **Import:** You can import a hook from the Hooks Inventory and assign it to the form field.
   - **Create New:** You can create a new hook and assign it to the form field.
   - **Modify:** You can modify an existing hook from the options available in the dropdown and assign it to the form field.

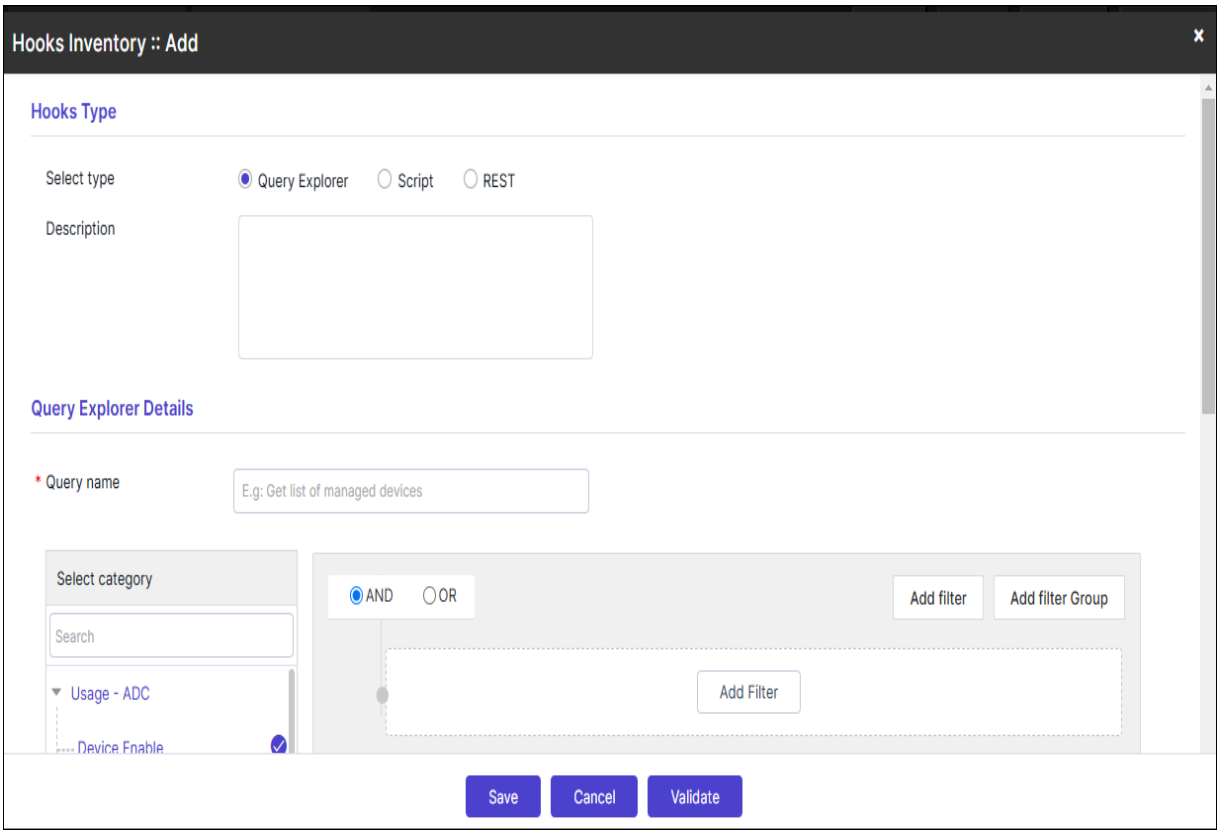
Creating a New Hook

1. Under the **Hooks**, click **Create New**.



   The **Hooks Inventory :: Add** window opens.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. Create a new hook as required and click **Save**.

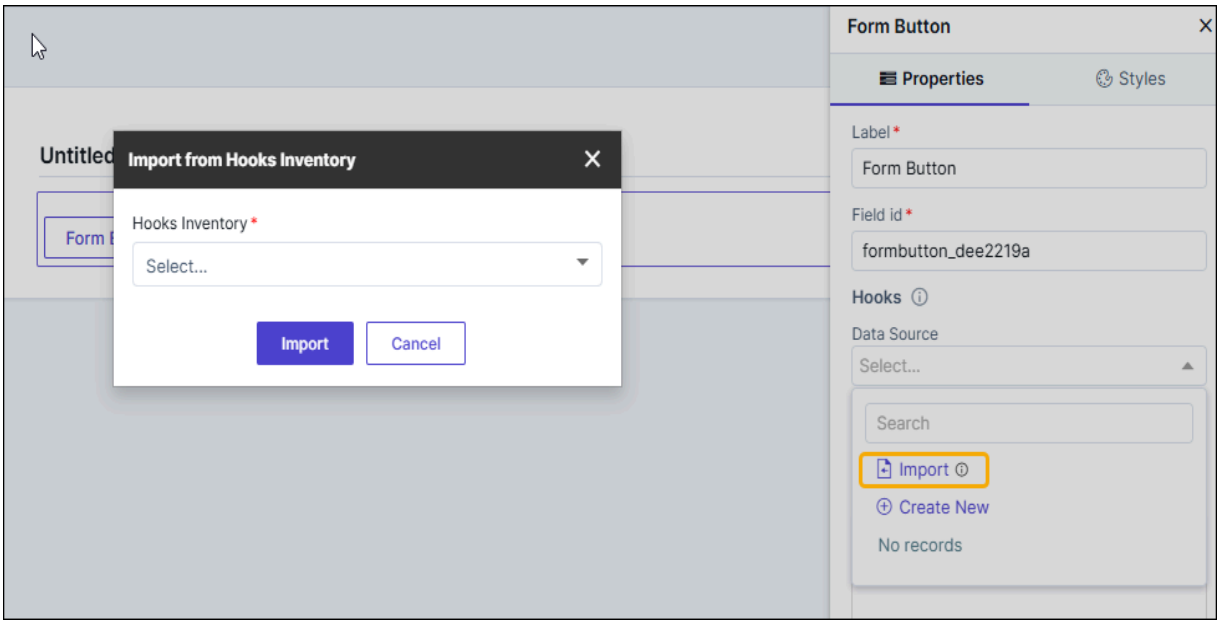
> **Note:** For more information on Hooks, refer to the Automation User Guide.

Importing Hooks from Inventory

1. Under the **Hooks** section, click **Import**.

   The **Import from Hooks Inventory** window opens with the list of hooks available in the inventory displayed in the dropdown.

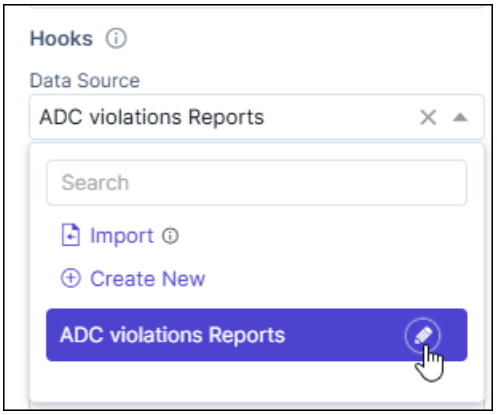
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. Select the required hook from the list and click **Import**.

Modifying a Hook

1. In the **Form** window, under the **Hooks** section, hover your mouse over the hook that is to be modified.
2. Click **(Edit)** icon.



The **Hooks Inventory :: Modify** window opens. You can modify the hook and assign it to the form field.

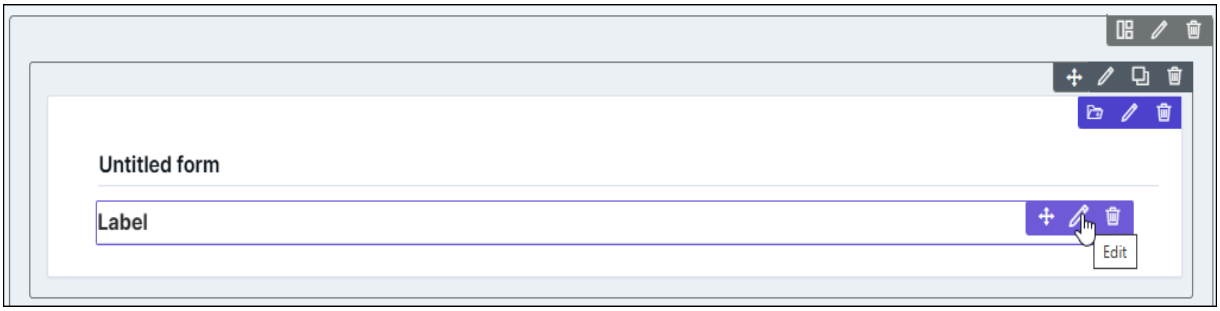
Note: For more information on Hooks, refer to the Automation User Guide.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Label

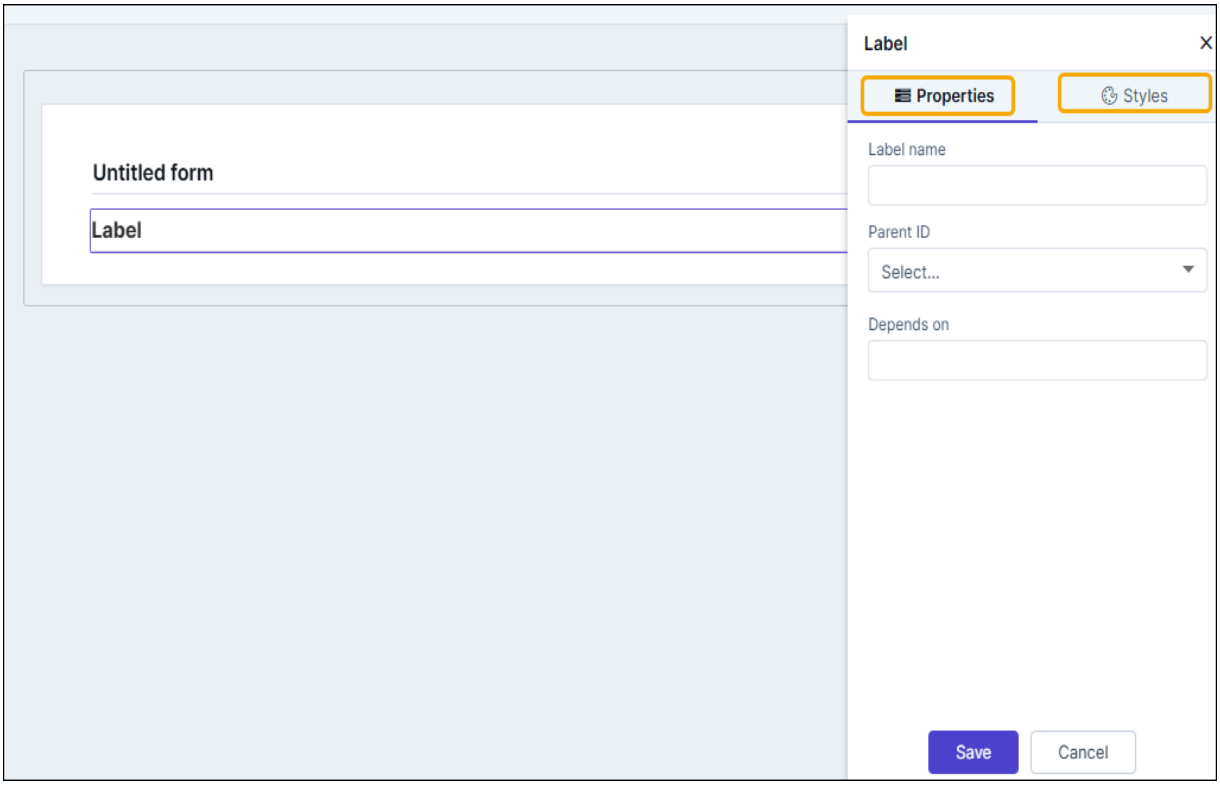
The Label component is a fundamental element within a form that serves the purpose of providing descriptive text or a title for form fields. It offers context and guidance to users, helping them understand the purpose of each field. It is mainly used to club together various properties within the tabular component.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Label** and drag it to the right on the empty new page.
   The Label component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Label Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Parent ID	It displays the ID generated for the tabular component added to the form. In the preview and published modes, the label won't appear on its own, but within the associated tabular component.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value**

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box **Note:** For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used

Editing Label Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Spacing	
Padding	Allows you to increase/decrease the space between the label name and border from the top, right, bottom, and left settings

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

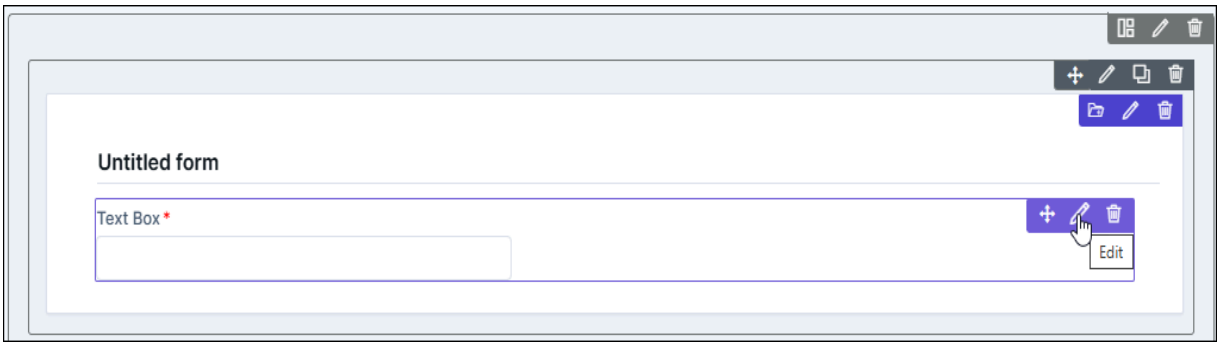
Field description for Styles (continued)

Fields	Description
Margin	Allows you to increase/decrease the border margins form the label name from the top, right, bottom, and left settings
Border Bottom	
Border Color	Allows you to define the color of the label border.
Border Width	Allows you to increase/decrease the width of the label border.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the to default, click icon.

Form Field - Text Box

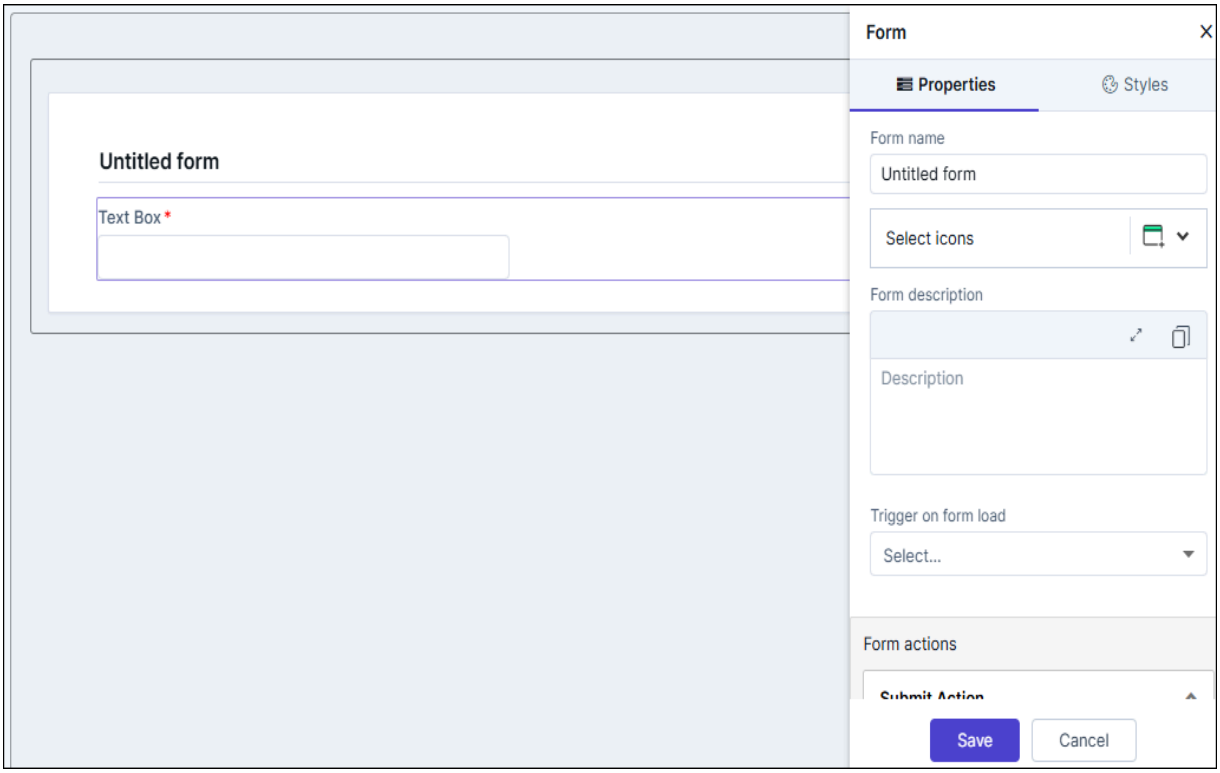
The Text Box component is a commonly used form input element that allows users to enter and edit single-line text data. It is ideal for capturing information like names, email addresses, or any other short text inputs.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Text box** and drag it to the right on the new page.
   The Text box component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Text Box Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Predefined Variable	Allows you to populate predefined values to appear in the text box field in the preview and published modes. The dropdown field contains values - **Hostname**, **Port**, and **Origin**.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Once the page is loaded, values can be edited in the field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: If the **Predefined Variables** are selected, then those preset values will take precedence over the text entered in the Values field.
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.

Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value**
	
	Examples are:
	
	`username:admin`
	`username:admin&password:admin`
	`(username:admin)|(password:admin)`

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	(username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box **Note:** For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.
Hidden	Allows you to hide the fields in the form. Used mainly in the configuration of hooks.
Add On Label	Allows you to add a label after the text box field. For example, if the field is used to specify number of days, you may add "Days" as the add on label
Range Selection	Allow you to enter the values as a range. The text box is split in two with "To" in between both boxes (similar to "From" and "To" range values that are commonly seen in forms).
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Text Box Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the field label name..
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the field label name..
Align	Allows you to align the field name to the left, center or right side of the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

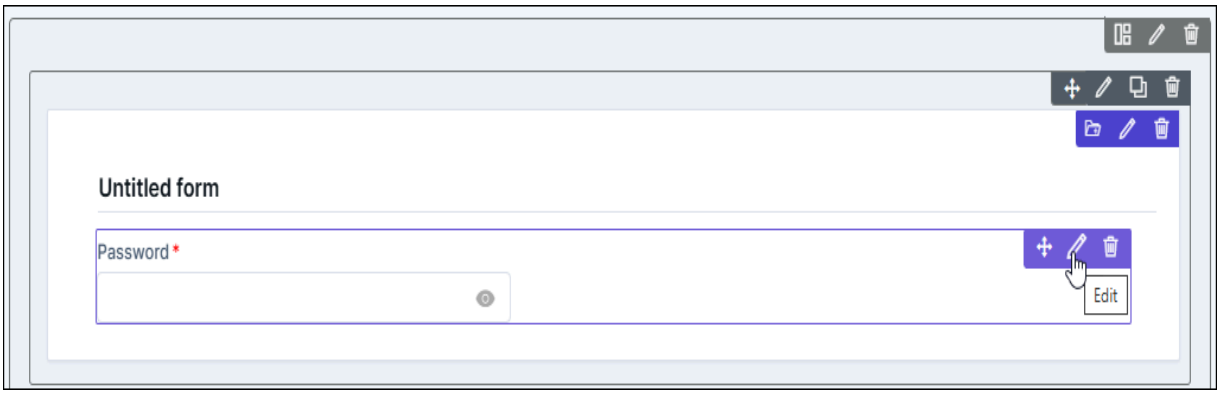
Field description for Styles (continued)

Fields	Description
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Form Field - Password

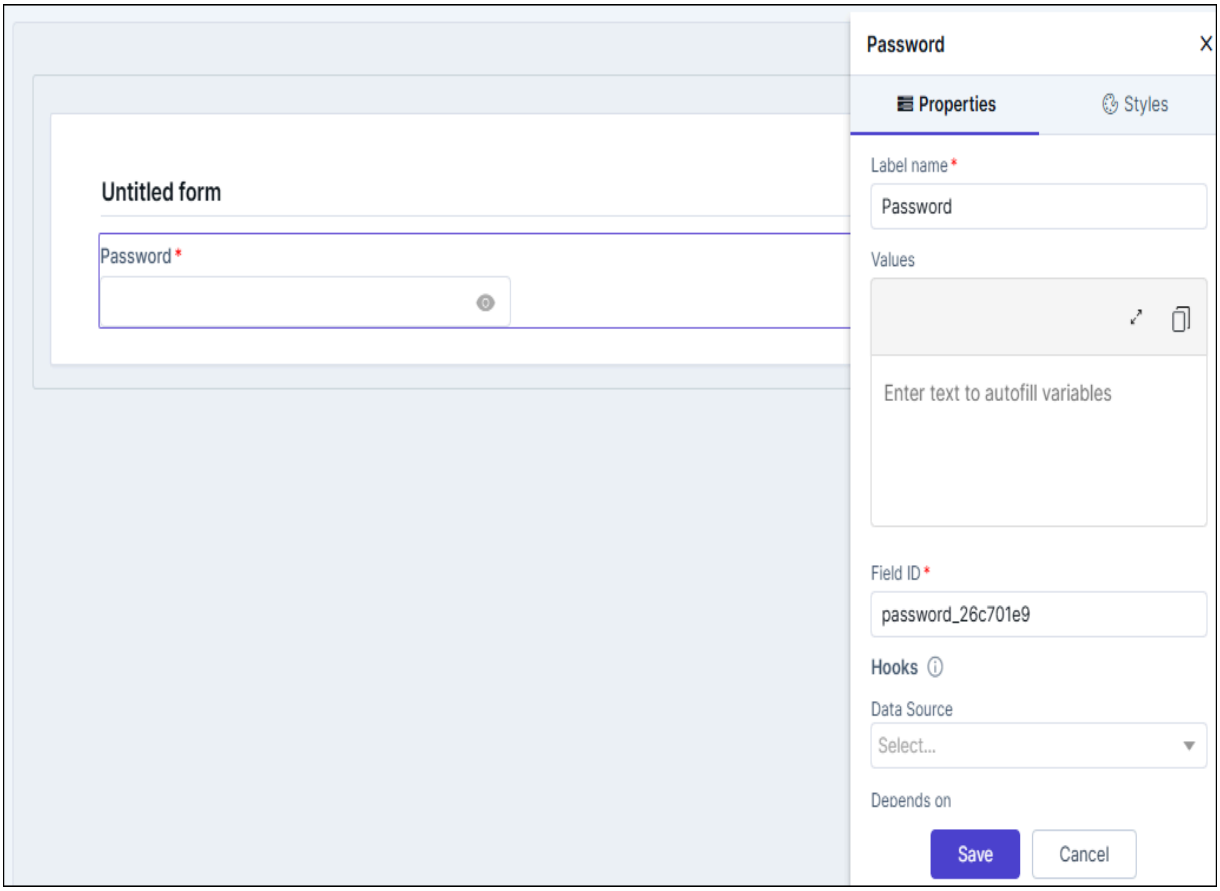
The Password field allows you to add a special text field that does not display text when the user types in it. Instead, the text typed by the user is displayed as an asterisk or a bullet to maintain confidentiality of sensitive information.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Password** and drag it to the right on the new page.
   The Password component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Password Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Once the page is loaded, values can be edited in the field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hidden	Allows you to hide the fields in the form. Used mainly in the configuration of hooks.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Password Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

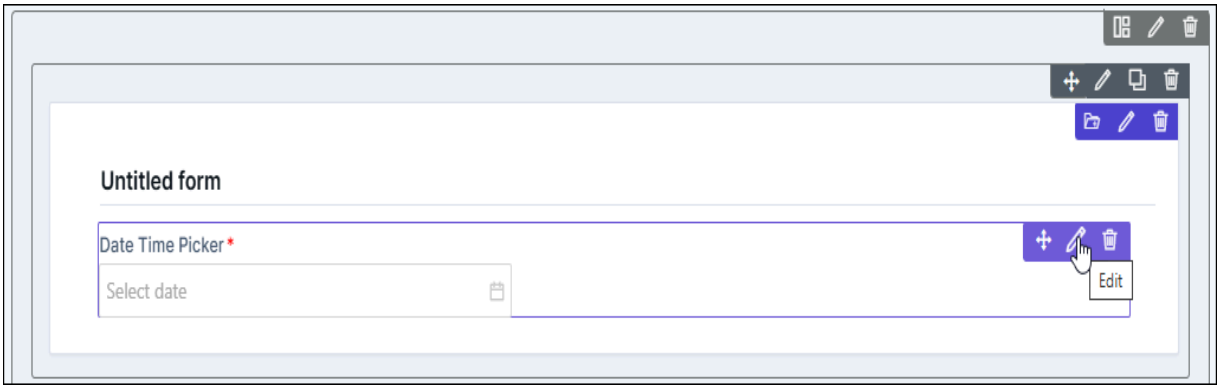
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Date Time Picker

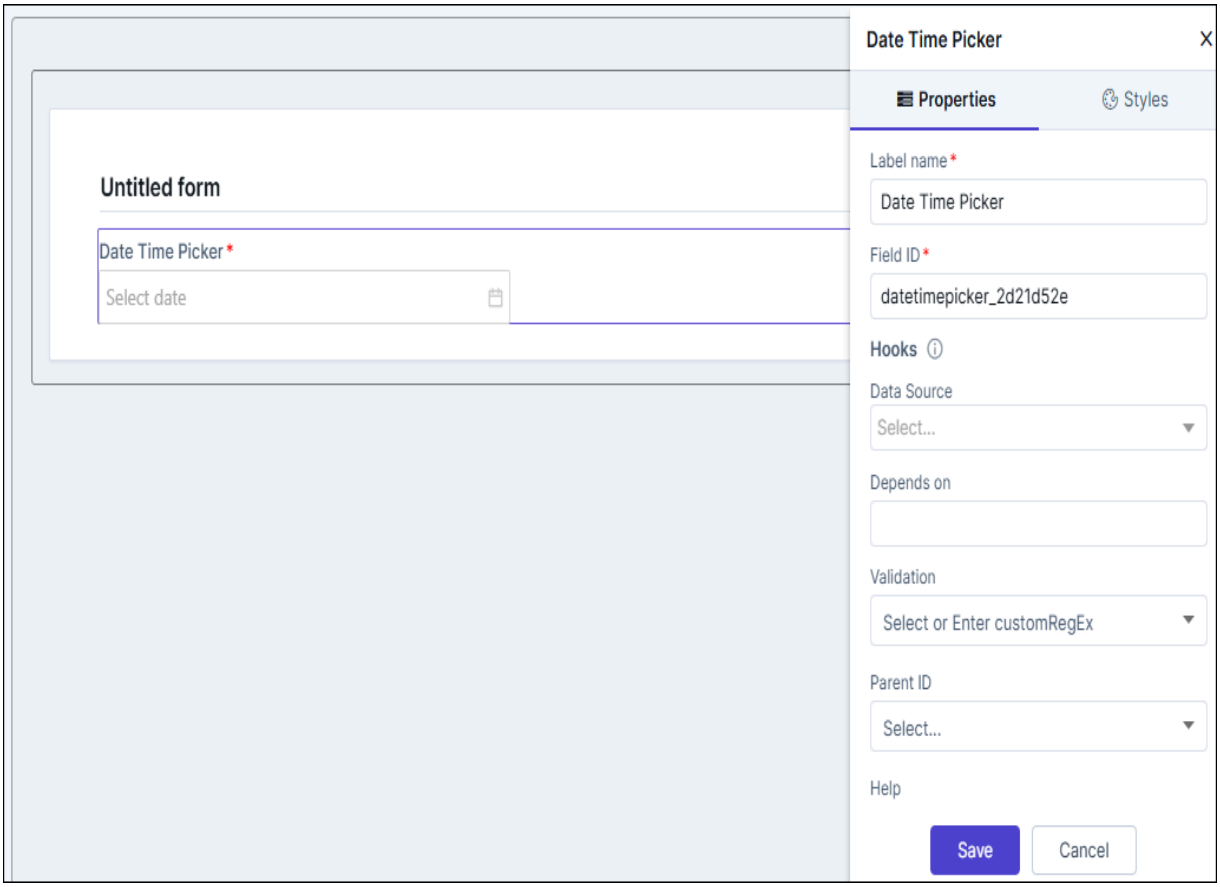
The Date Time Picker component enables users to select date and time values using a user-friendly calendar interface. It simplifies the process of inputting dates and times accurately.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Date Time Picker** and drag it to the right on the new page.
   The Date Time Picker component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Date Time Picker Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.

Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value**
	
	Examples are:
	`username:admin`
	`username:admin&password:admin`
	`(username:admin)|(password:admin)`
	`(username:admin)&(password:admin)`
	`((username:admin)&(password:admin))|(device:adc)`
	`(username:admin)|(password:startsWith.a)`
	`(username:admin)|(username:admin,user 1) //for multi -select box`
	
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used

Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.
Range Selection	Allow you to enter the values as a range. The text box is split in two (Start Date & End Date) with "To" in between both boxes.
Show Time	Enables the selection of time (Hours: Minutes: Second) along with the date.
Save / Cancel buttons	To update the changes, click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	To undo the changes, click **Cancel**.

Editing Date Time Picker Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

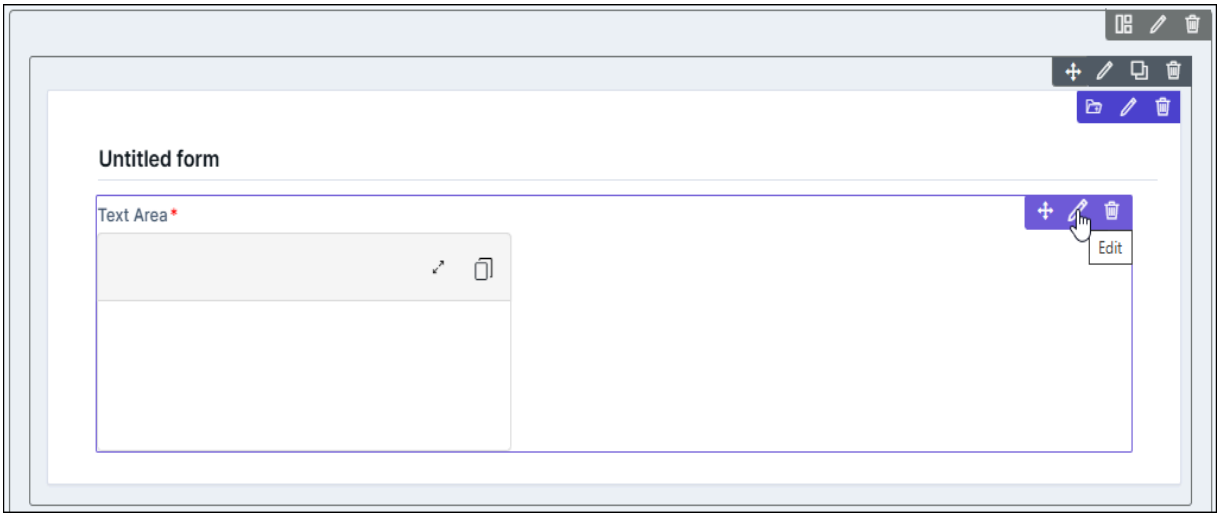
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the field label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the field label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Form Field - Text Area

The Text Area component is used to collect multi-line text data, such as comments or descriptions. It allows users to enter longer text inputs and supports text wrapping. The Text Area once published on a form has default options to expand and copy text.

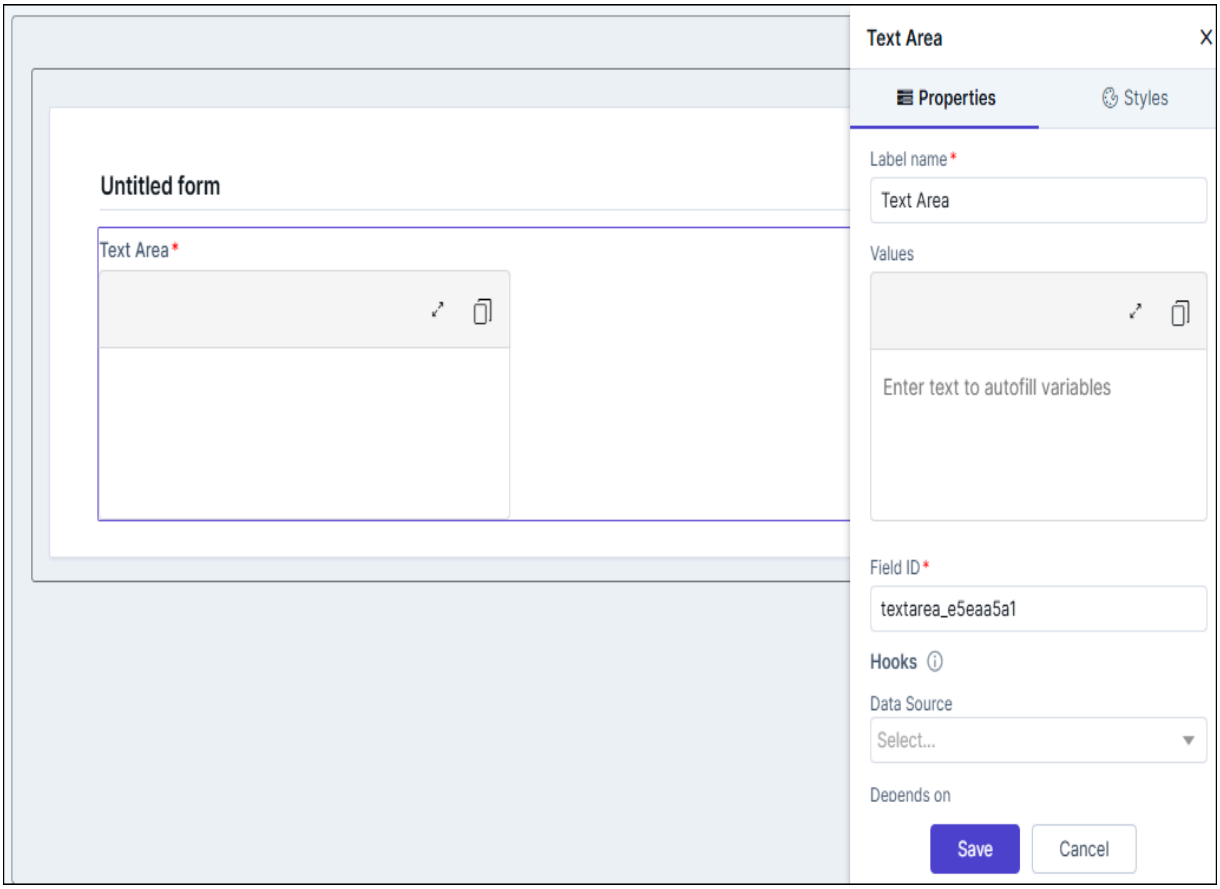
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Text area** and drag it to the right on the new page.
   The Text area component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Text Area Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Once the page is loaded, values can be edited in the field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hidden	Allows you to hide the fields in the form. Used mainly in the configuration of hooks.
File Upload	Allows you to add a File upload icon **(Upload)** icon in the Text Area. The following files can be uploaded: .p7b, .p7c, .pfx, .p12, .der, .csv, .txt, .crt, .pem, .csr, .cer, .key.
Show Header	Allows you to hide the label name of the text area.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Text Area Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

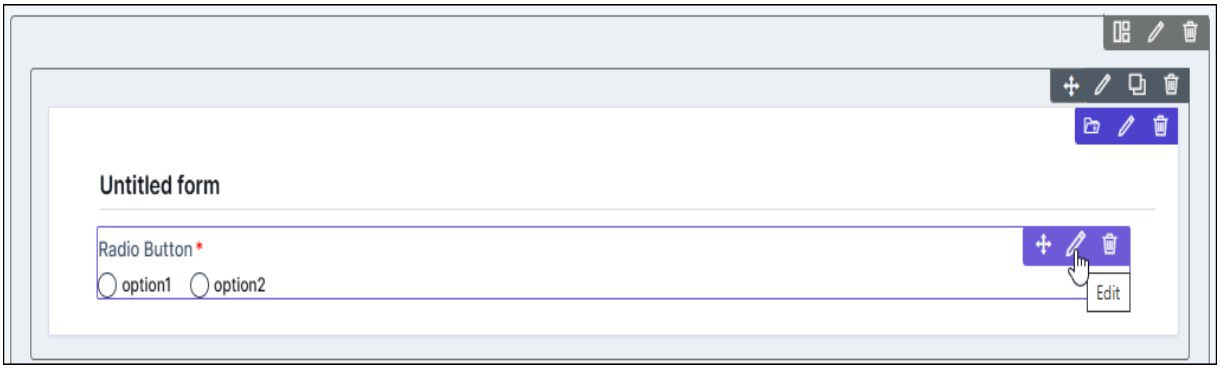
Field description for Styles (continued)

Fields	Description
	To restore the settings to default, click icon.

Form Field - Radio Button

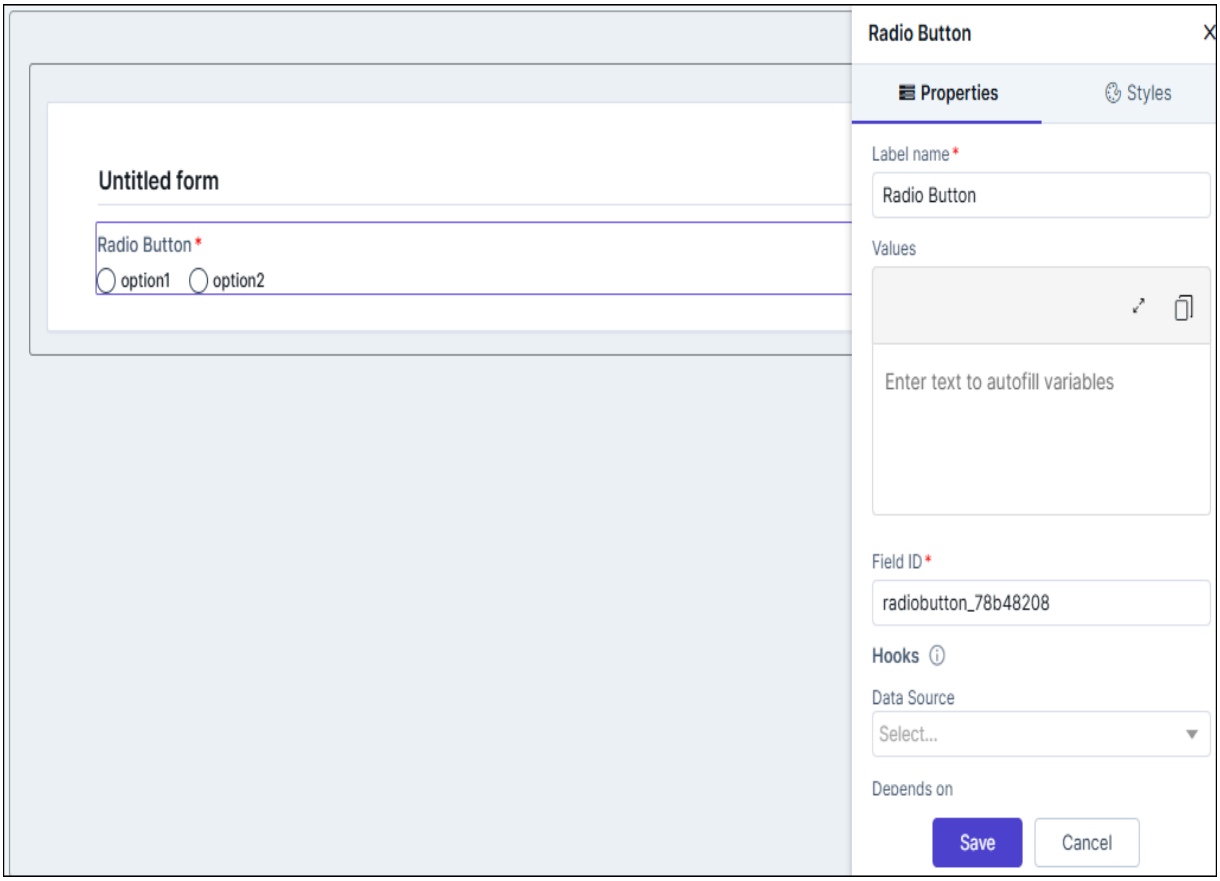
The Radio Button component provides a set of options from which users can select a single choice. It presents a list of mutually exclusive options, and users can choose only one.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Radio button** and drag it to the right on the new page.
   The Radio button component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Radio Button Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Enter the comma separated values; each value is considered a radio button option.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hidden	Allows you to hide the fields in the form. Used mainly in the configuration of hooks.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Radio Button Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

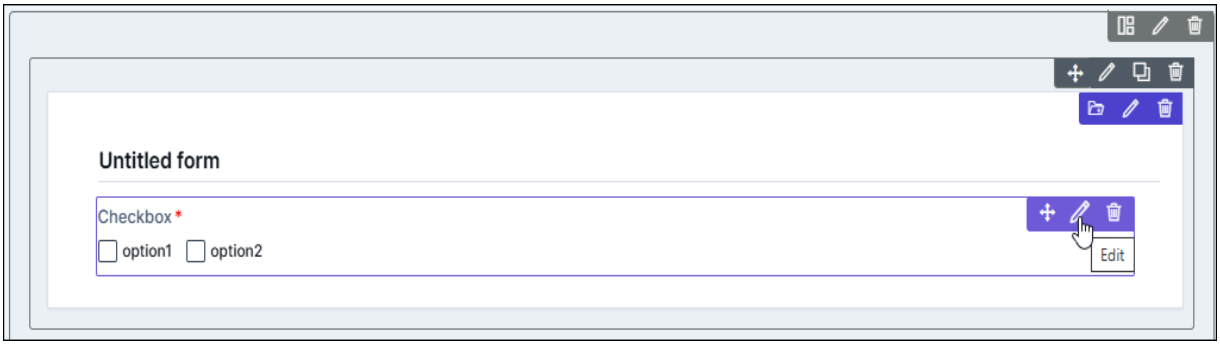
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Checkbox

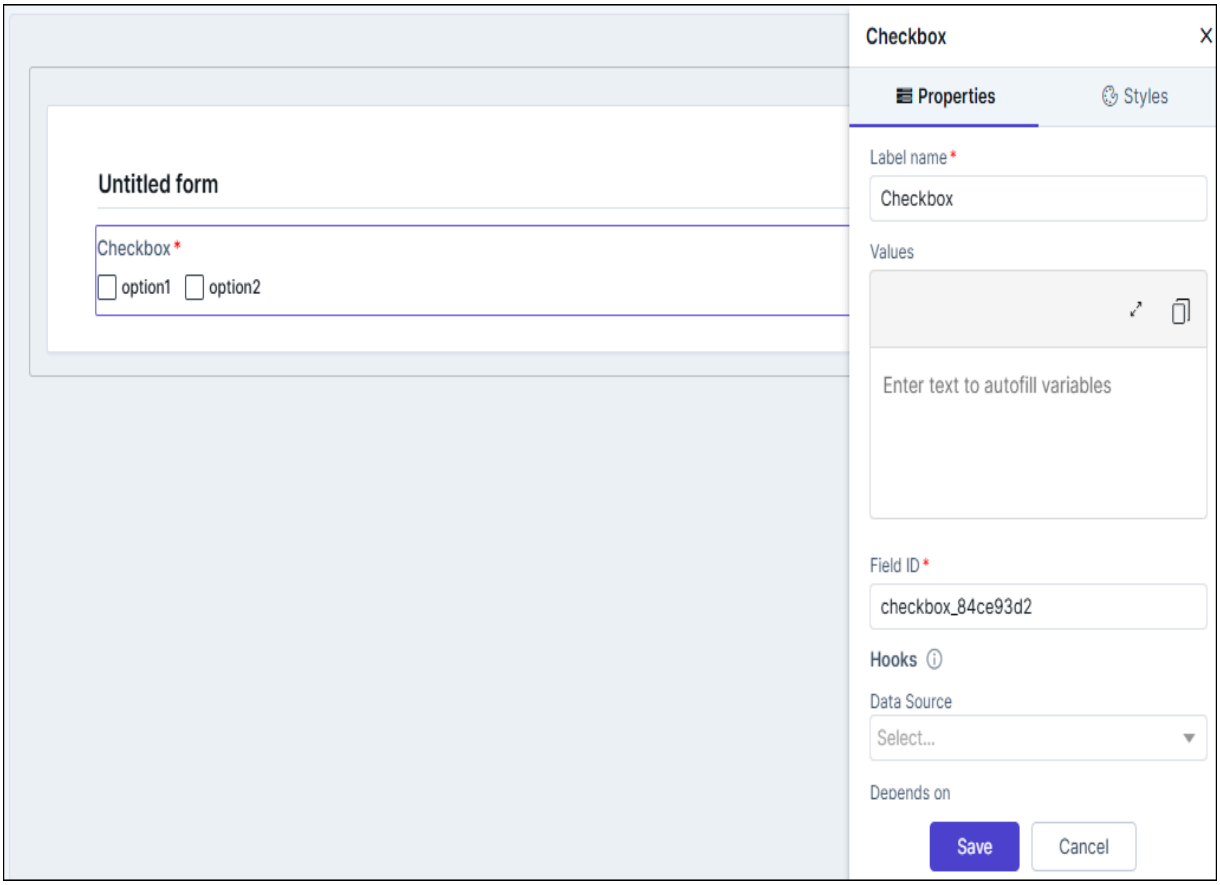
The Checkbox component offers a list of options, and users can select multiple choices simultaneously by checking the corresponding checkboxes.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Checkbox** and drag it to the right on the new page.
   The Checkbox component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Checkbox Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Enter the comma separated values; each value is considered a checkbox option.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Convert to Switch	Allows you to convert the component to a toggle switch (with enable/ disable functionality). The values entered for checkbox are no longer valid.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Checkbox Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

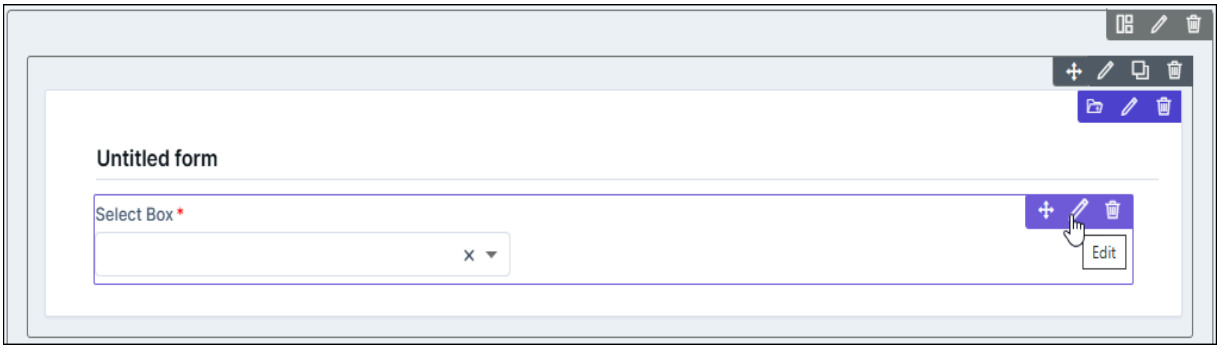
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Select Box

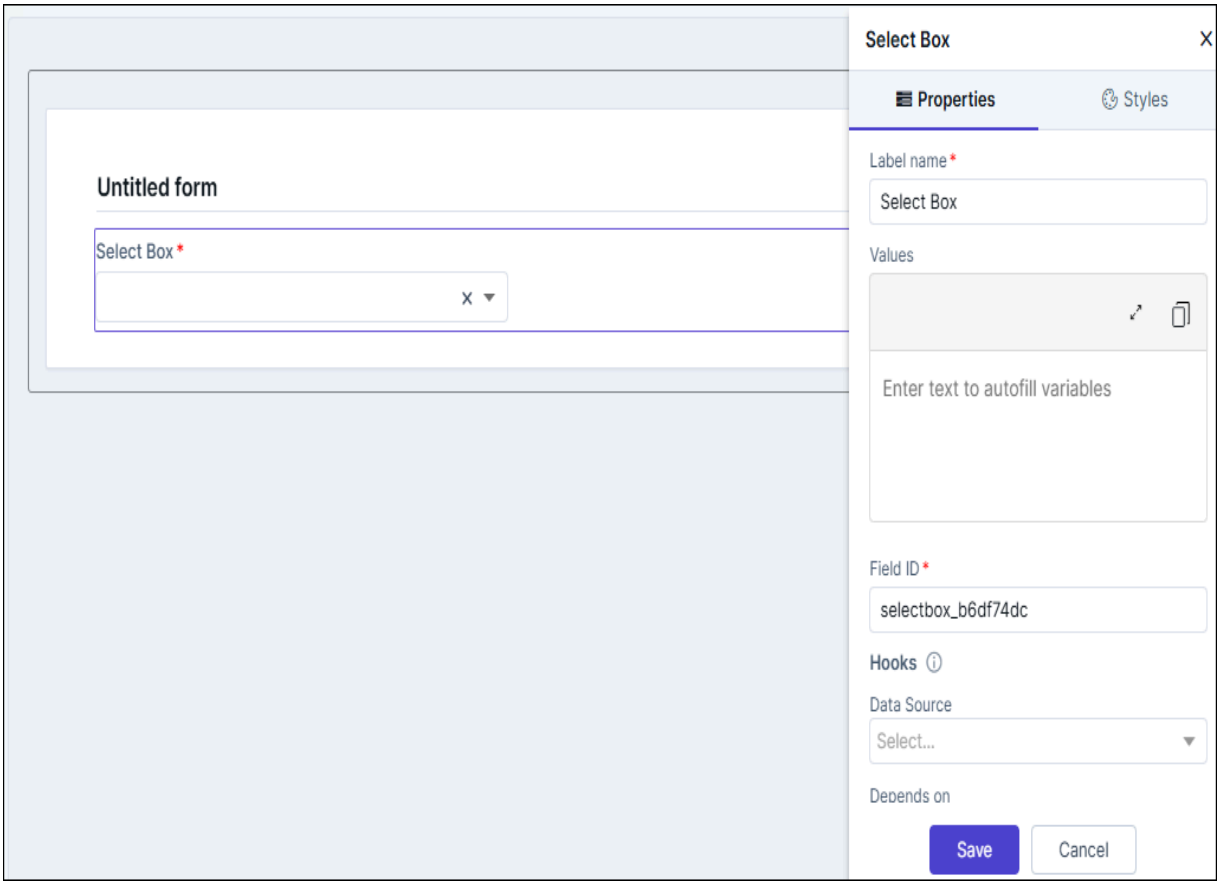
The Select Box component, also known as a dropdown menu, presents a list of options in a drop-down format. Users can select one option from the list or create his own option that gets added to the existing list.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Select Box** and drag it to the right on the new page.
   The Select Box component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Select Box Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Enter the comma separated values; each value is considered a dropdown option. A default value to appear on page load can be configured using hooks.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Create option	Allows dynamic value creation. After the page loads, users can type a value into the field and select the prompt below the field to add the value.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Select Box Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

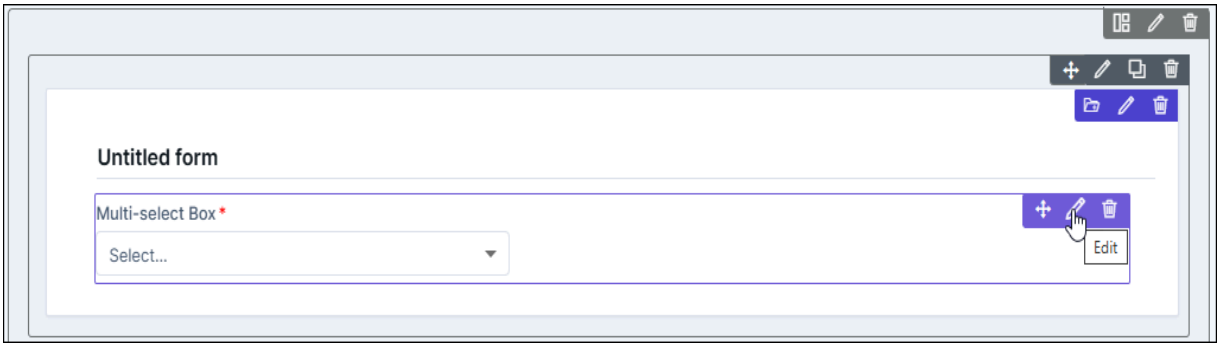
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Multi-select Box

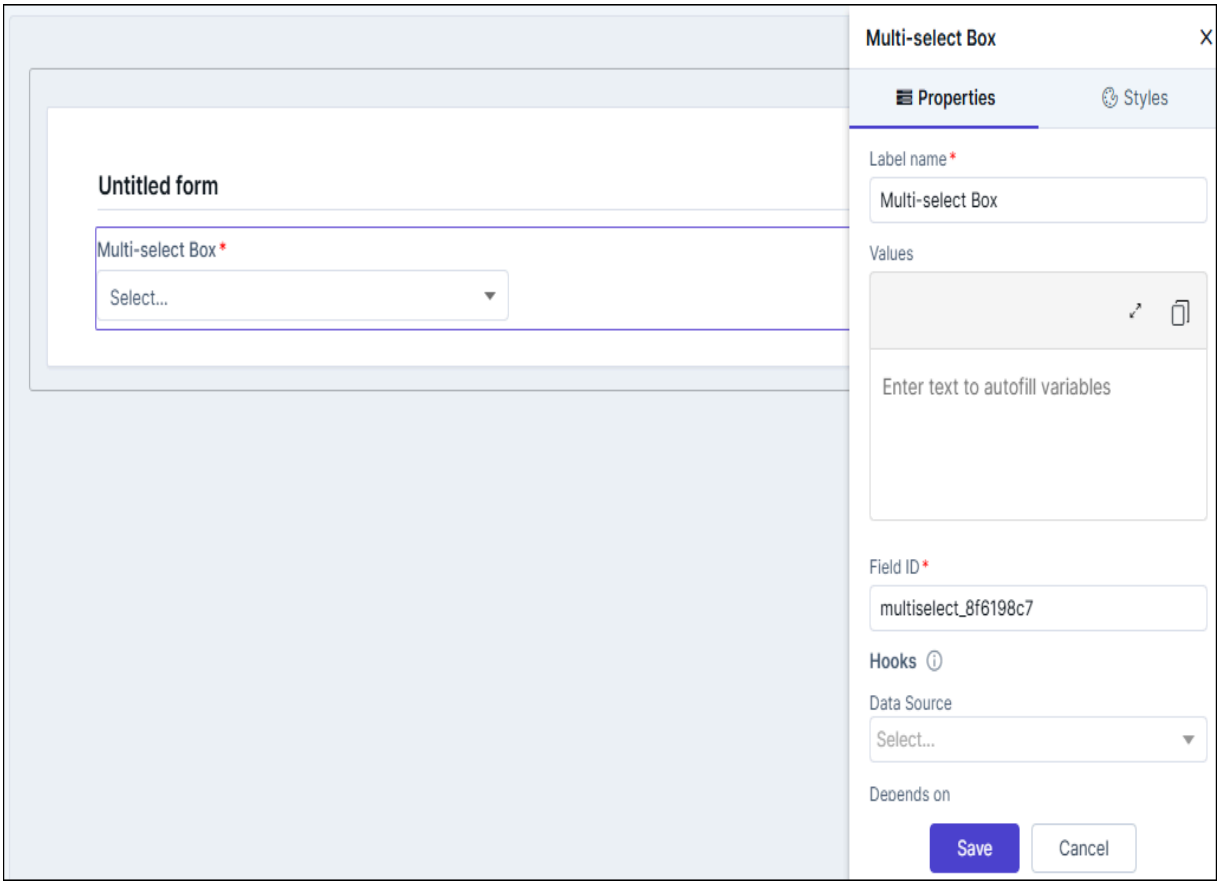
The Multi-Select Box component is similar to the Select Box but allows users to select multiple options from the list.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Multi-select Box** and drag it to the right on the new page.
   The Multi-select Box component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Multi-Select Box Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Enter the comma separated values; each value is considered a dropdown option. A default value to appear on page load can be configured using hooks.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Create option	Allows dynamic value creation. After the page loads, users can type a value into the field and select the prompt below the field to add the value.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Multi-Select Box Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

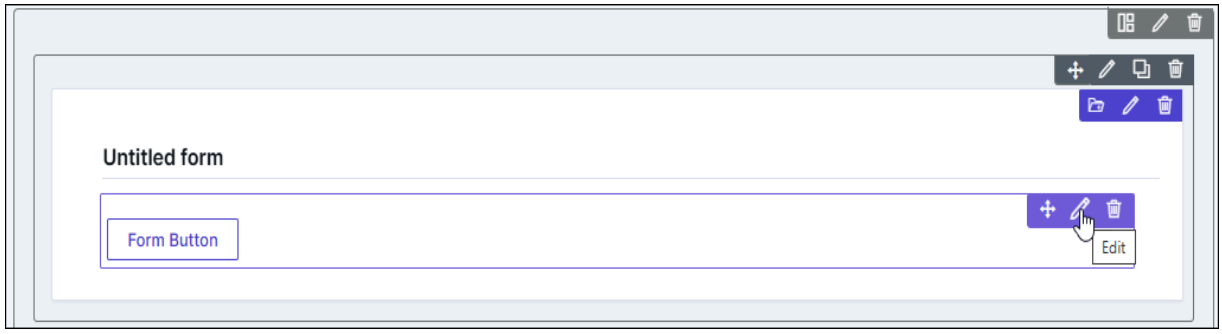
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Form Field - Form Button

The Form Button component represents action buttons within a form, such as "Submit," "Cancel," or other custom actions. These buttons execute specified actions before the form is submitted. For instance, they can be used to populate fields below the button based on the data entered in the fields above it.

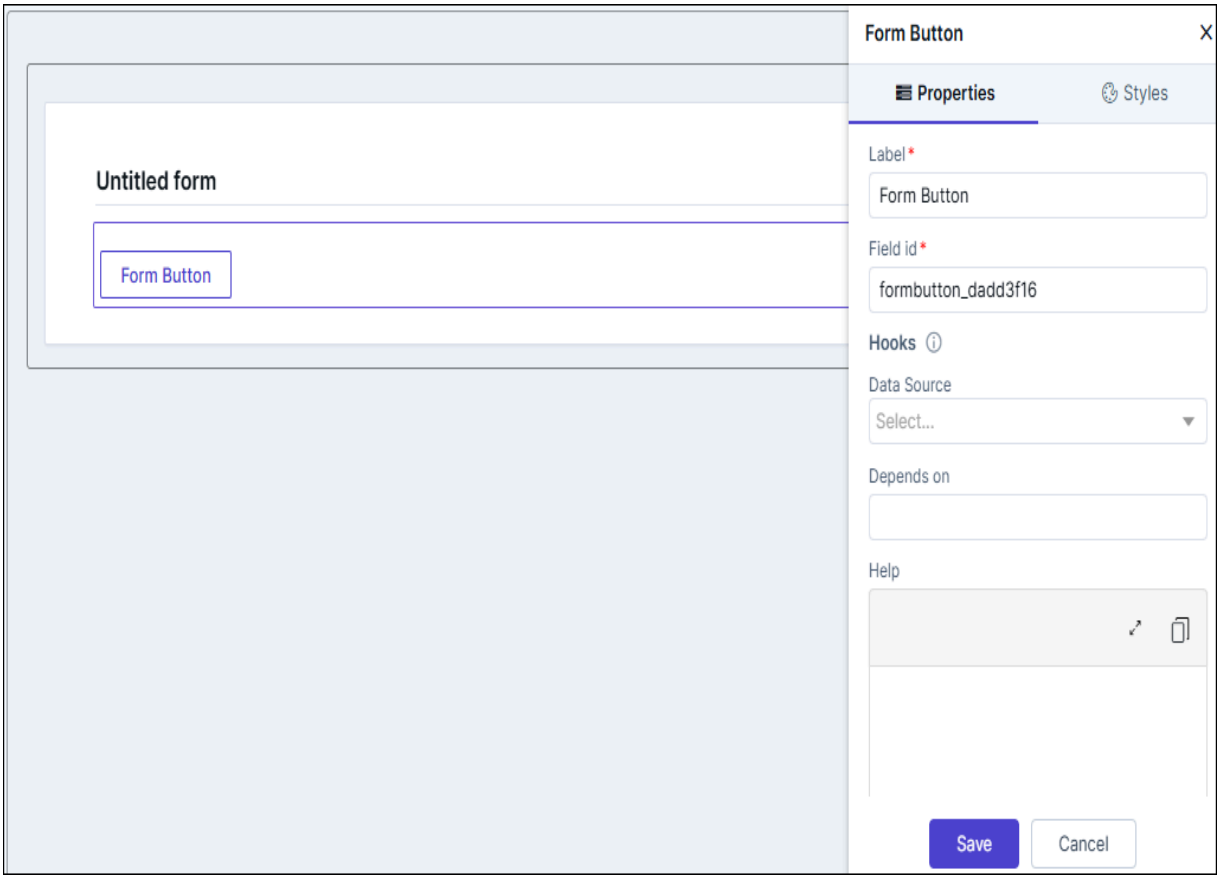
1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Form Button** and drag it to the right on the new page.

   The Form Button component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Form Button Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.

Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value**
	
	Examples are:
	`username:admin`
	`username:admin&password:admin`
	`(username:admin)|(password:admin)`
	`(username:admin)&(password:admin)`
	`((username:admin)&(password:admin))|(device:adc)`
	`(username:admin)|(password:startsWith.a)`
	`(username:admin)|(username:admin,user 1) //for multi -select box`
	
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Form Button Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Layout	
Background - Color	Allows you to change the background color of the button (behind the text).
Border - Color	Allows you to change the border color of the button.
Border - Style	Allows you to change the border type to dotted, dashes etc. Select the appropriate values from the dropdown.
Border - Width	Allows you to change the border width (top, bottom, right, left) by pixels.
Set Box shadow on hover	Allows you to enable the shadow when hovering over the button.
Field Label	

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Styles (continued)

Fields	Description
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size component (label) name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the field text.
Spacing *(area inside the button)*	
Padding	Allows you define the spacing between the button text and the button borders from the top, left, bottom, and right.
Margin	Allows you define the spacing of button layout in the form, from the top, left, bottom, and right.
Background - Color	Allows you define the background color (which is between the button and the form borders) from the top, left, bottom, and right.
Border - Color	Allows you define the background border color (which is between the button and the form borders) from the top, left, bottom, and right.
Border - Style	Allows you define the background border color (which is between the button and the form borders) from the top, left, bottom, and right.
Border - Width	
Layout Spacing *(area outside the button)*	
Padding	Allows you define the spacing between the button and the background area from the top, left, bottom, and right.
Margin	Allows you define the spacing of button layout in the form, using the top, left, bottom, and right pixel adjustments.
Field Size	
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

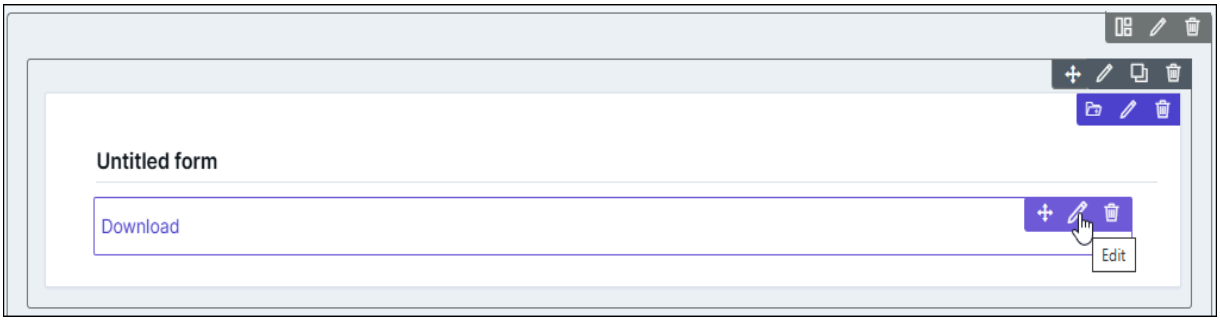
Field description for Styles (continued)

Fields	Description
	To restore the settings to default, click icon.

Form Field - Download

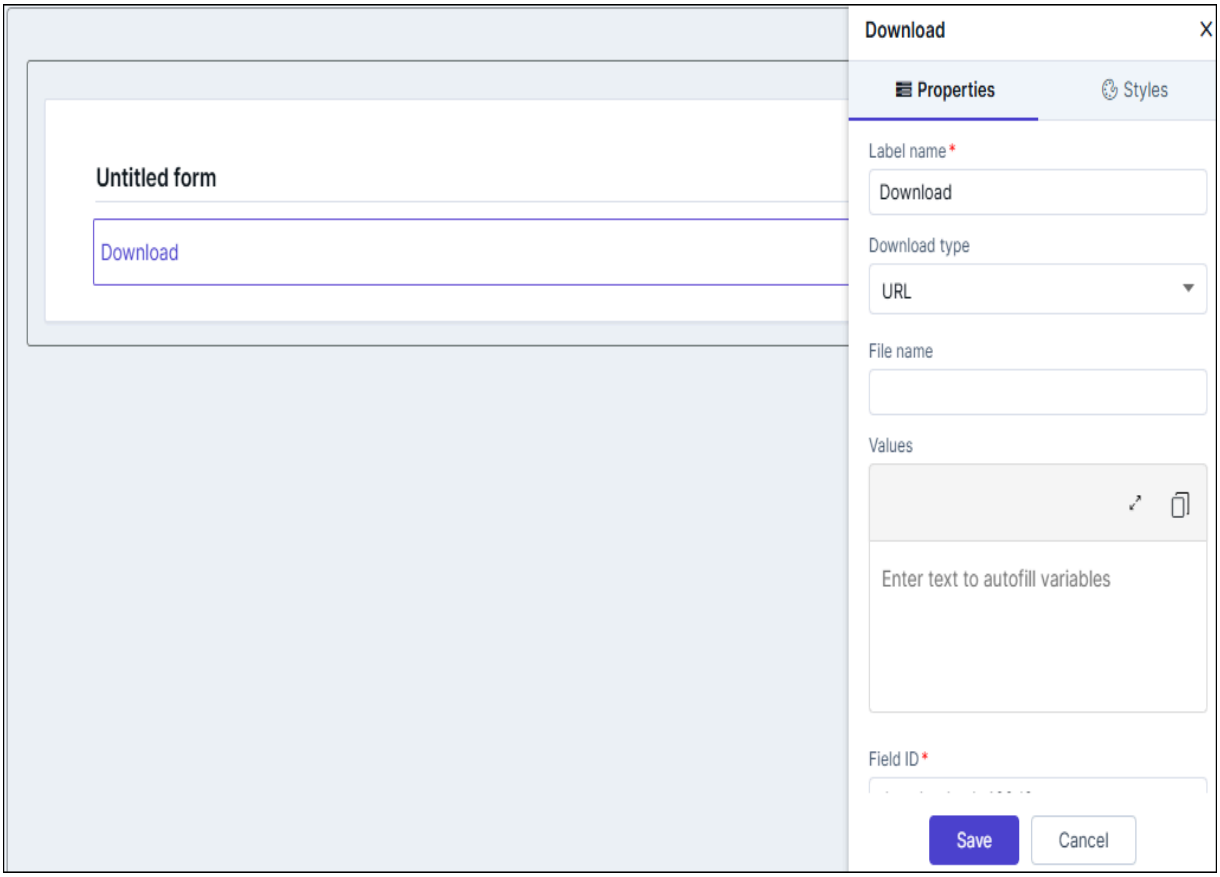
The Download component enables users to download files or resources directly from the form. It provides a link to download files like documents, or other attachments.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Download** and drag it to the right on the new page.
   The Download component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Download Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Download type	Select the content to download - **CSV**, **URL**, **File**.
File name	Enter the name for the downloaded file.
Values	Allows you to define a values that will appear on the downloaded CSV, URL, and File. When downloading as CSV, enter comma separated values. Each value is downloaded in a separate cell of the CSV file.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	When downloading as URL, enter the repository location (example a dropbox location).
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc)

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	(username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box **Note:** For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Download Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Button Size	Allows you to increase/decrease size of the download button.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

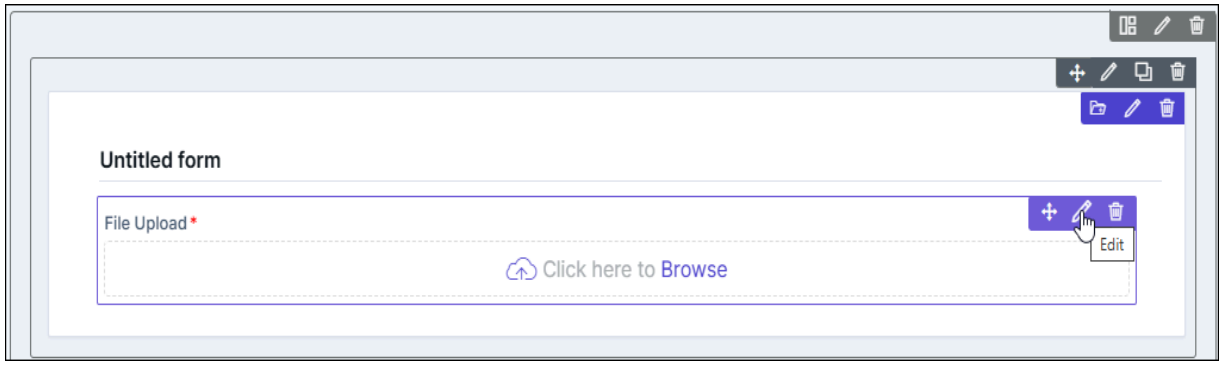
Form Field - File Upload

The File Upload component allows users to upload files into the form. Users can attach relevant files to the form fields.

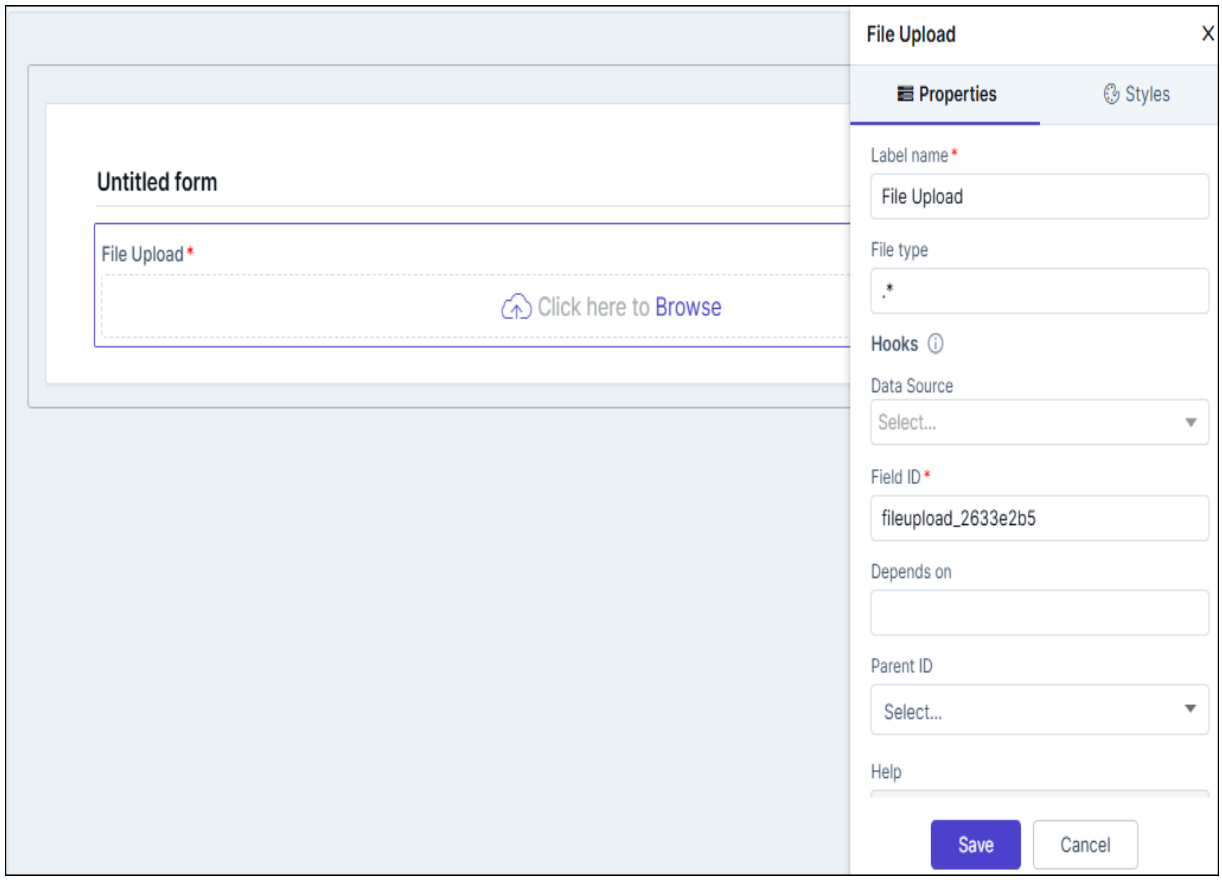
1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **File Upload** and drag it to the right on the new page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The File Upload component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing File Upload Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
File Types	Allows you to define the types of files that can be uploaded. Enter values with . (dot) and file extension, such as **.csv**. To allow multiple file types enter the value as **.***.
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Password Required	Enable the toggle button and enter the password for the uploaded file in the **Password required for file types** field provided.
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box **Note:** For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Editing File Upload Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Header Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Form Field - Script Editor

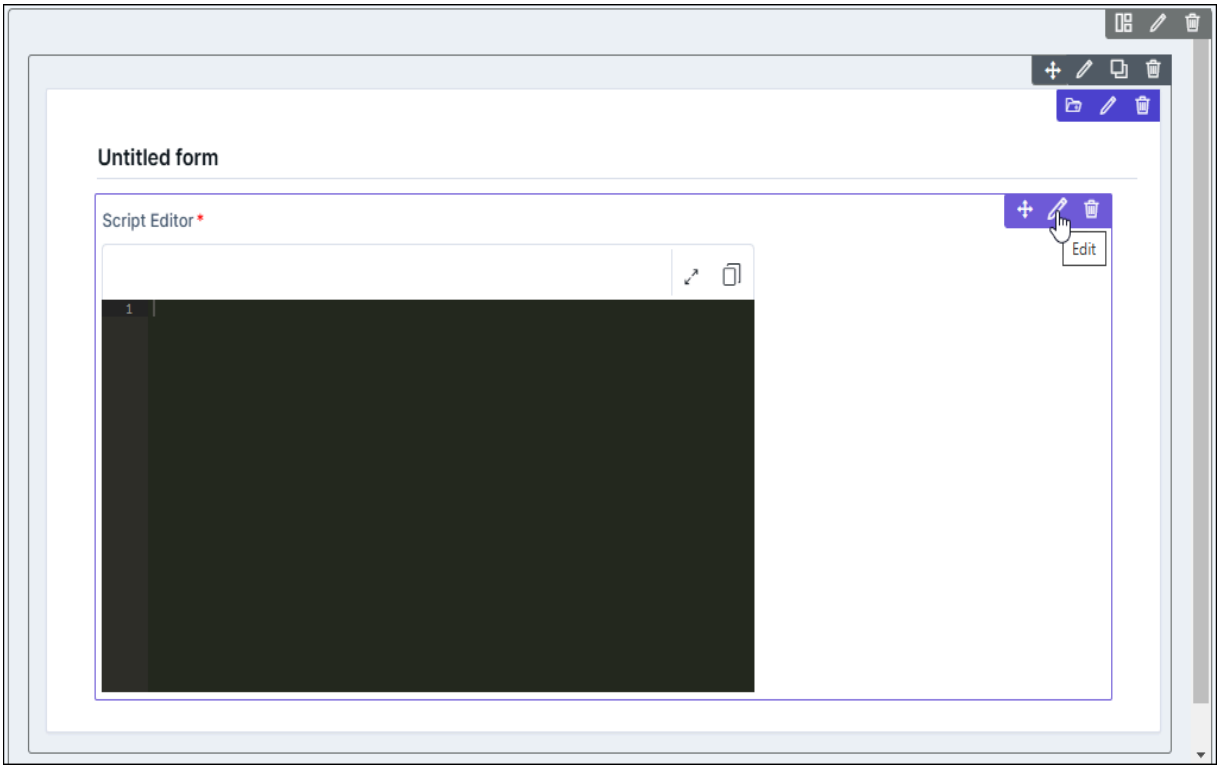
The Script Editor component enables users to write and edit scripts, such as JavaScript, Python,JSON, XML, YAML, and many more within the form. It offers a text editor with syntax highlighting for script editing.

Editing Text Area Properties

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.

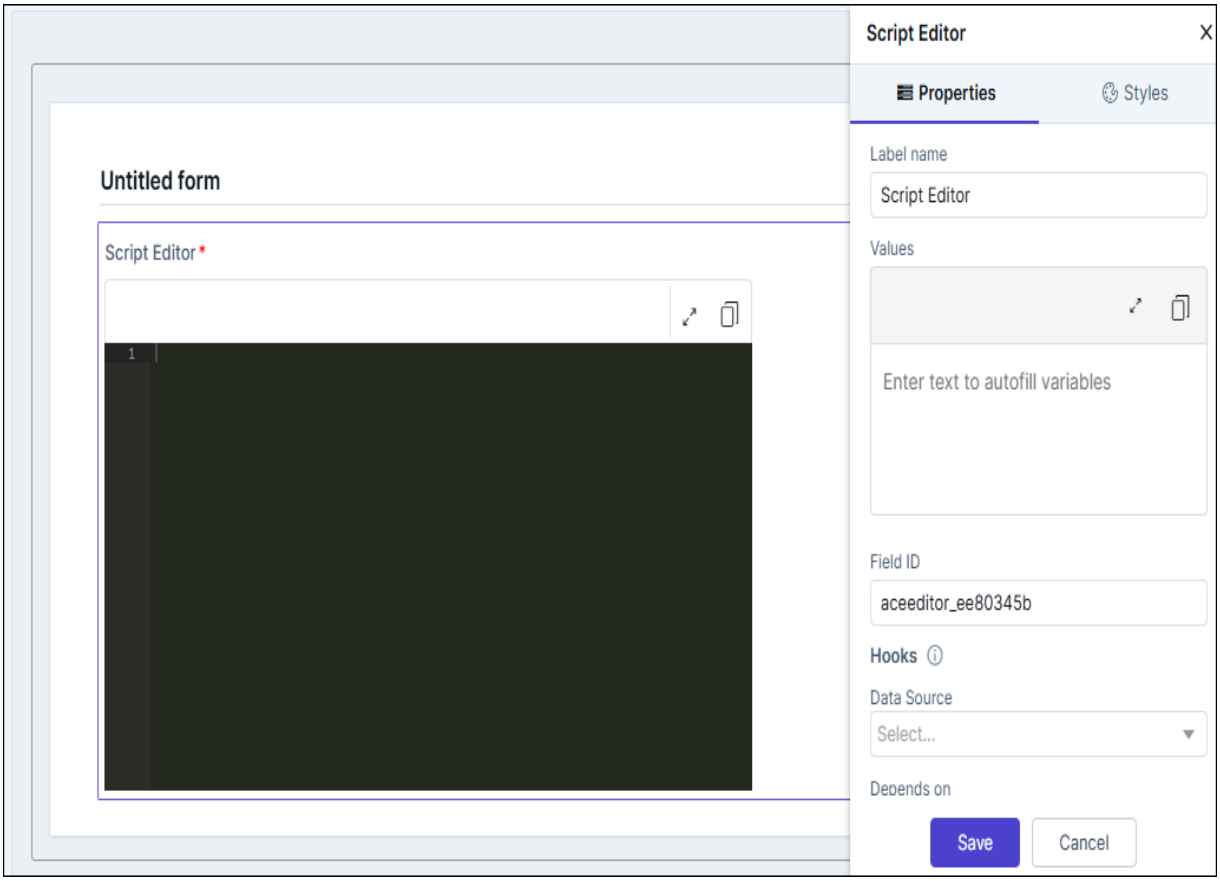
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Script Editor** and drag it to the right on the new page.
   The Script Editor component is added in the form, with icons to *Move*, *Edit*, and *Delete*.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Script Editor Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Values	Allows you to define a values that will appear on the page load in the preview and published mode. Once the page is loaded, values can be edited in the field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks / Data Source	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Validation	The Validation property provides users with the ability to define validation rules for form fields. These rules ensure that the entered data meets specific criteria, maintaining data accuracy and integrity. Users have the privilege either to use the validations already available or create custom regex validation. Example: sample validation for *No special characters and spaces are allowed except('-', '_') and name should not start with special characters. Maximum length mismatches. Maximum Length - 255* Validation Regex would be: `^[^_-][a-zA-Z_-]{1,255}$`
Parent ID	Allows you to assign a Parent ID for fields in a table. Select the relevant tabular component (Field ID) against which the form field must be mapped to. **Note:** This is used only when defining a tabular element in the form.
Help	Allows you to add Help text/tool-tip for the field.
Mandatory	Allows you to define this field as Mandatory or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Global variable	The "Global Variable" property enables users to reference and use global variables within form fields. These variables provide dynamic values that can be utilized in calculations or conditional logic within the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties (continued)

Fields	Description
Hidden	Allows you to hide the field in the form. Used mainly in the configuration of hooks.
File Upload	Allows you to add an option for uploading files to the ace editor field.
File Download	Allows you to add an option for downloading files from the ace editor field. The following fields are displayed when the toggle is enabled. • **File Name** - Enter the name for the file to be saved when downloaded. • **Download Type** - Select the file type for downloads (TXT, YAML, SH, CSV, PDF, JSON.)
Theme	Allows you to select between a light or dark theme for the ace editor.
Mode	Allows you to select the script editor mode from the drop down menu.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Script Editor Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Styles (continued)

Fields	Description
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

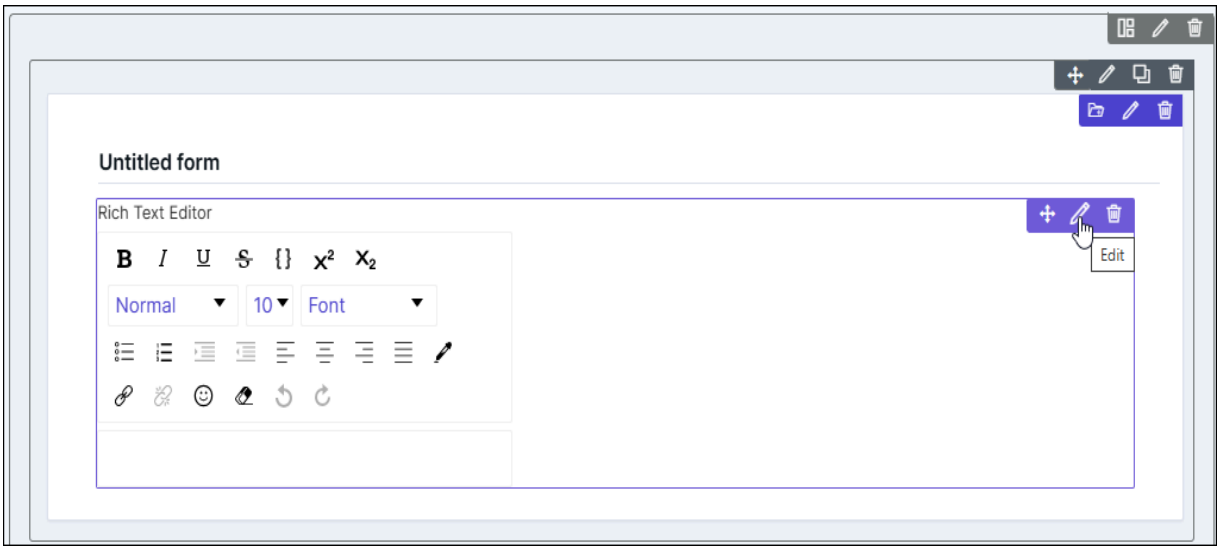
Form Field - Rich Text Editor

The Rich Editor component is a versatile text editor that offers various formatting options, such as bold, italic, underline, bullet points, and more. It is suitable for composing rich-text content within the form. The Rich Text Editor field allows you to add a text editor to your page with the following features:

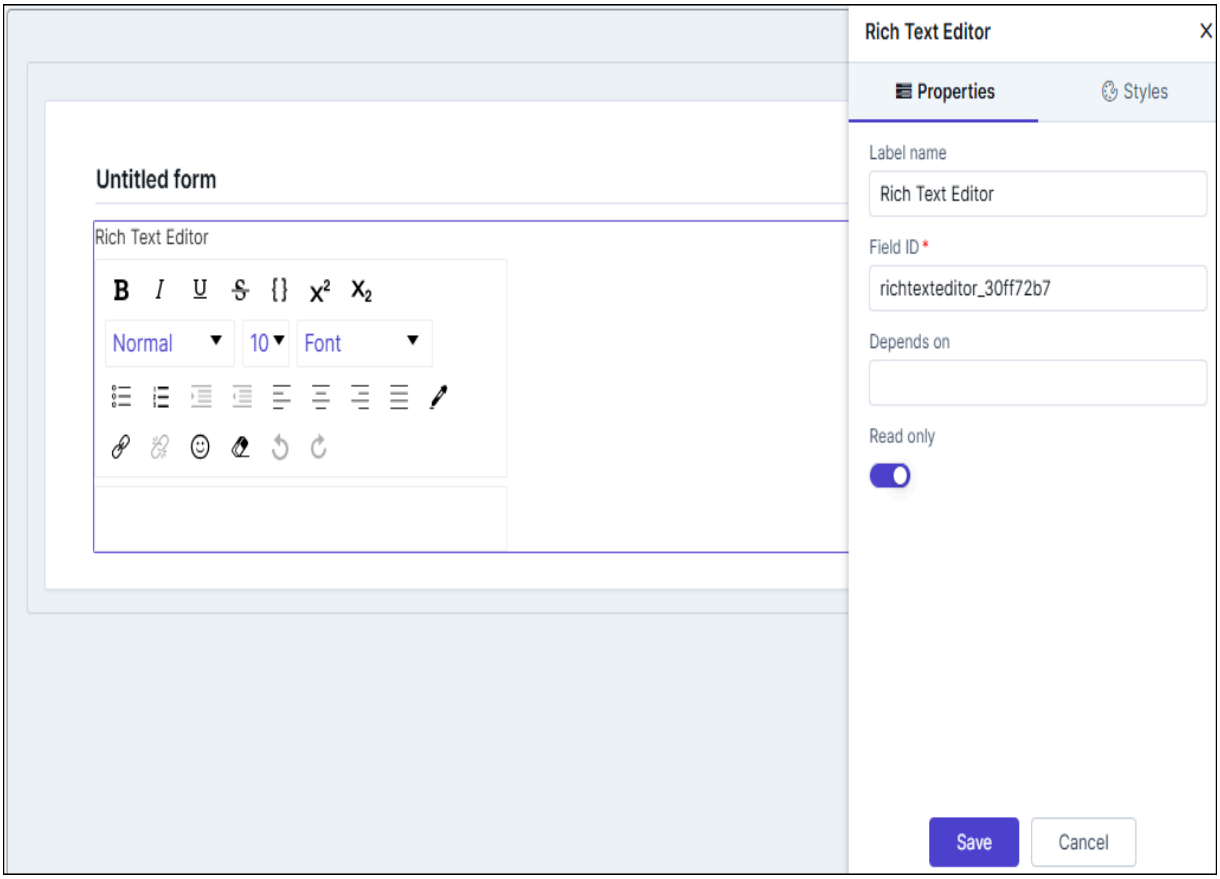
- Provision to define the font type and size, add font effects, highlight text in color, and add bulleted and numbered points.
- Provision to enable monospace.
- Provision to insert superscript and subscript.
- Provision to add links to external/internal pages.
- Provision to embed an internal/external page within the text editor.
- Provision to indent and align the text as per requirement.
- Provision to upload files to the page.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Rich Text Editor** and drag it to the right on the new page.
   The Rich Text Editor component is added in the form, with icons to *Move*, *Edit*, and *Delete*.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Rich Text Editor Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties

Fields	Description
Label name	Enter a valid name for the component.
Field ID	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value** Examples are: username:admin username:admin&password:admin (username:admin)|(password:admin) (username:admin)&(password:admin) ((username:admin)&(password:admin))|(device:adc) (username:admin)|(password:startsWith.a) (username:admin)|(username:admin,user 1) //for multi -select box **Note:** For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used
Read only	Allows you to define this field as read only or otherwise. If marked as Read Only, a red crossed circle is displayed when you hover your mouse over the read only field and does not allow you to enter any information.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Editing Rich Text Editor Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

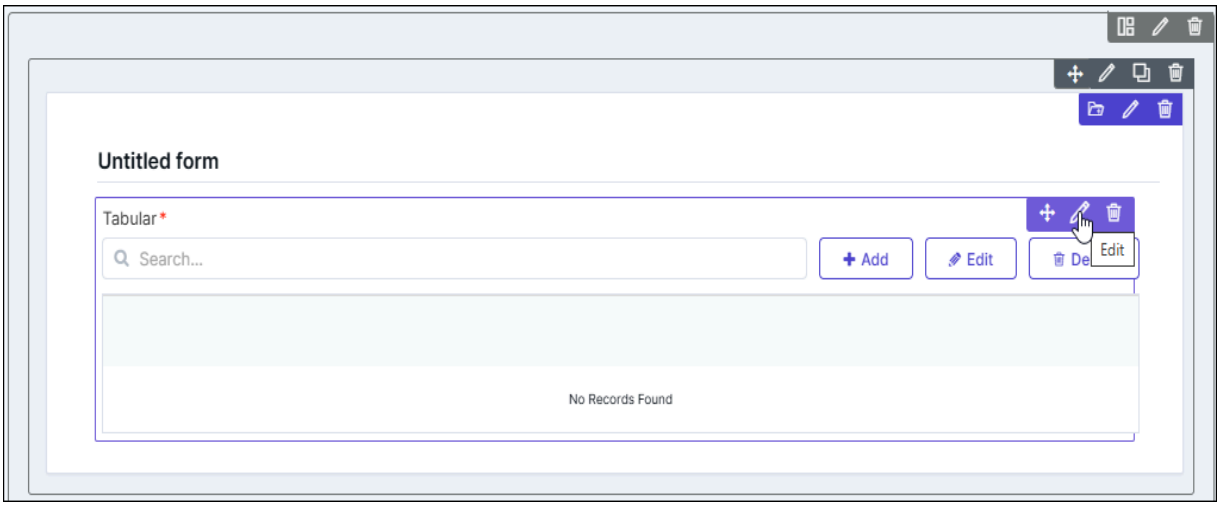
Fields	Description
Font	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component/label name.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Widget Background color	Allows you to choose a background color for the widget.
Widget Header - Padding	Allows you to adjust the top, right, bottom, and left padding of the widget in the form.
Field Size	Allows you to define the size of the form field as Small, Medium, or Large.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Form Field - Tabular

The Tabular component represents data in a tabular format, allowing users to view and interact with structured data. It presents data in rows and columns, providing a clear organization.

1. To add a Tabular element to the form first design a form by adding form fields to the design canvas.
2. Drag and drop the **Tabular** field in the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



3. Hover your mouse over a field, for example **First Name**, and click **(Edit)** icon.
   The **Text Box** pop-up window opens with the **Properties** tab open by default.
4. In the **Text Box** pop-up window, under **Properties**, select the **Parent ID** as Tabular.
   The field, **First Name**, is added to the Tabular.
5. Associate the Tabular's field ID as the **Parent ID** for other fields that are to be added to the form.

> **Note:** When the Tabular field within a Form Group only the form fields present within that group can be added to that group's Tabular.

6. Click **Preview**.
   The Tabular opens in a new tab.

> **Note:** Any form fields not associated with the Tabular field will be displayed outside the Tabular.

7. To add data in the Tabular component, click Add.
   The **User Details's fields** window opens with the fields of the tabular displayed.
8. In the **User Details's fields** window, enter the values for the form fields.
9. Click **Save**.
   The values are added to the Tabular.

> **Note:** The **User Details's fields** window remains open for the user to add more fields.

10. To close the **Tabular's fields** window, click **Cancel**.
11. To edit a row in the Tabular, select the row to be edited.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The **Edit** button is enabled.

12. Click **Edit**.

> **Note:** The **Edit** button is enabled only when you select exactly one row in the tabular. Selecting zero or more than 1 row will disable the **Edit** button.

13. In the **User Details's fields** window, modify the form field(s).
14. Click **Update**.

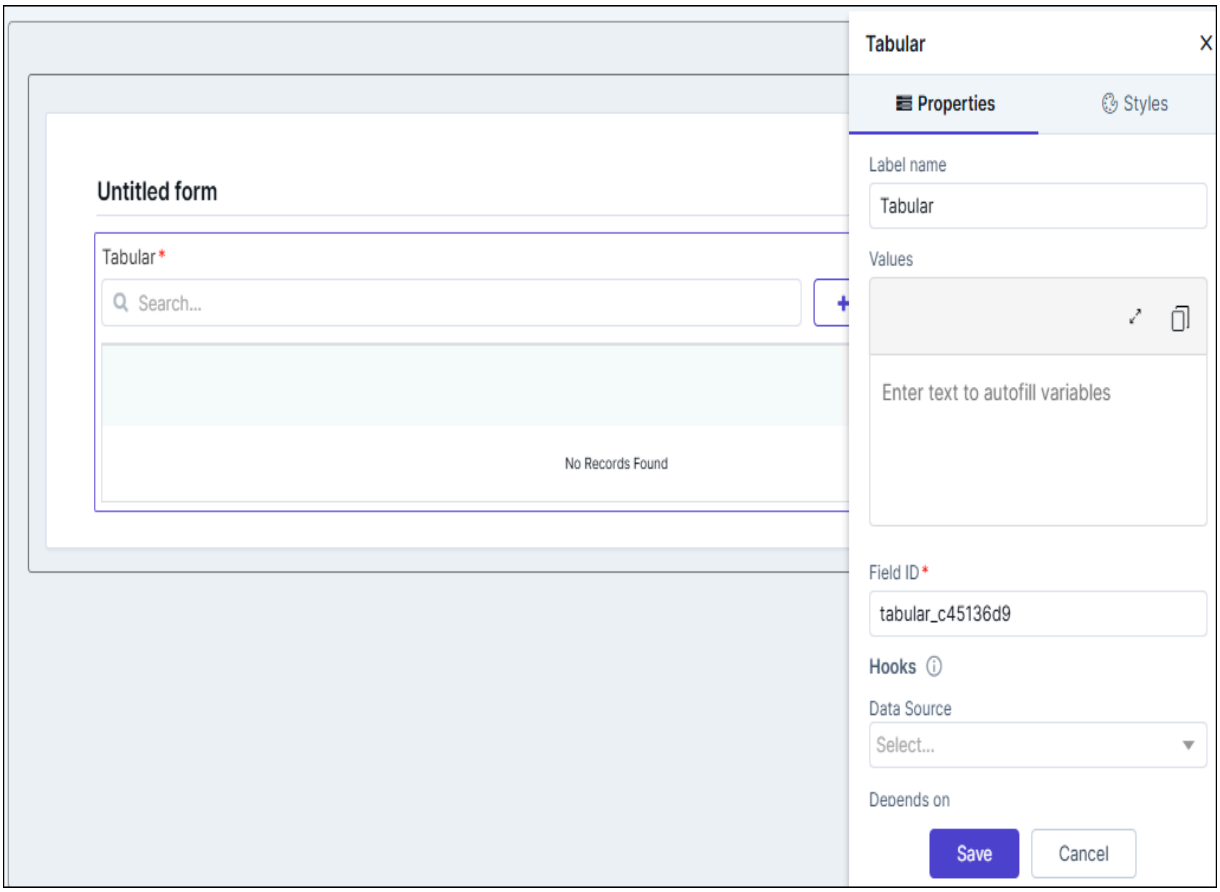
    The Tabular's fields window closes.
15. To delete a row in the Tabular, select the row to be deleted.
16. Click **Delete**.

> **Note:** The **Delete** button is enabled only when you select a single or multiple rows.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Tabular - Properties

1. To configure the **Tabular**, hover your mouse over the field and click **(Edit)** icon.



The **Tabular** pop-up window opens.

2. In the **Tabular** window, under **Properties**, enter or select the required fields.

Field descriptions for Properties

Field	Description
Label name*	Enter a valid **Label name** for the form field. For example, Select Date and Time.
Values	Allows you to define field **Values** relevant to a form field. • Values can be delimited by a comma. • Characters not allowed: '&' and '|'.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Field ID*	An auto-generated Id for the field, this can be edited. The IDs are useful while creating hooks. • Allowed characters: '-' and '_' • Character Length: Min (1), Max (50).
Hooks	Allows you to assign an appropriate hook for this field. For example, a hook to fetch the current date. • Select a hook that is relevant to the form 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. **Note:** For more information on Assigning Hooks to Form Fields, click here.

Depends on	The "Depends On" property enables rendering a field based on the selected data of another field. It establishes dependencies between form elements, dynamically updating the form layout based on user interactions. For example, if a user wants to render a field to enter email id, only if the username matches admin then the depends on value will be like `username:admin` . Ideally the syntax would be **field_id:value**
	
	Examples are:
	`username:admin`
	`username:admin&password:admin`
	`(username:admin)|(password:admin)`
	`(username:admin)&(password:admin)`
	`((username:admin)&(password:admin))|(device:adc)`
	`(username:admin)|(password:startsWith.a)`
	`(username:admin)|(username:admin,user 1) //for multi -select box`
	
	Note: For AND operations only single ampersand (&) is used and for OR operations only single pipe (|) is used

Help	Allows you to add Help text/tool-tip for the field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Profile Access	Allows you to select the profile from the dropdown to restrict form field display within a template based on the relevant profile.
Mandatory	Allows you to define this field as **Mandatory** or otherwise. If marked as mandatory, an asterisk (*) is displayed next to the field name in the form.
Grid Universal Controls	Allows you enable/disable the following features in the Tabular: • **Search** • **Pagination** • **Actions**. **Note:** For more information on **Grid Universal Controls**, click here.
**: Mandatory fields*	

3. Click **Save**.

Tabular - Grid Universal Controls

This option in the Tabular allows you to enable/disable the following features:

- **Search** - You can turn off the toggle for Search if you don't want the Search bar to be displayed in the Tabular. Enabling the Search bar allows you to search for a particular item in the Tabular quickly.
  - Search toggle is on and the search bar is displayed.
  - Search toggle is off and the search bar is not displayed.
- **Pagination** - You can turn on the toggle for Pagination to display the number of records that can be viewed on a page. Enabling Pagination allows a page with high volumes of data to load faster.
  - Pagination toggle is on and the record view is displayed.
  - Pagination toggle is off and the record view is not displayed.
- **Actions** - You can turn off the Actions toggle if you don't want the Action buttons to be displayed in the Tabular. Enabling action buttons such as Add, Edit, and Delete allows you to perform these actions and modify the Tabular quickly.
  - Actions toggle is on and the action buttons are displayed.
  - Actions toggle is off and the action buttons are not displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring the Tabular with Hooks

1. Design a page.
2. From the **Build** > **Form** section, drag and drop the following form fields to design a form:

Field Type	Label Name	Field Values	Parent ID
Text Box	Vendors	A10, F5, Citrix	-
Text Box	Device Name	-	Tabular
Text Box	IP Address	-	Tabular

3. Drag and drop a Tabular field to the form.
4. To configure the **Tabular**, hover your mouse over the field and click **(Edit)** icon.
   The **Tabular** pop-up window opens with the **Properties** tab open by default.
5. In the **Tabular** window, under **Properties**, select an appropriate hook for getting the device list.

> **Note:** You can also create a hook that meets your requirements. For more information on assigning hooks to form fields, click here.

6. Click **Save**.
7. **Preview** the form.
   The form opens in a new tab.
8. In the **Vendors** field, select an option from the dropdown.
9. To trigger the hook and get the device list based on vendor and auto-populate the form, click above the Tabular field.
   The Tabular is populated with the device list fetched from the hook.

Configuring the Tabular - Styles

1. For more options to customize the Tabular, in the **Tabular** window, click **Styles**.

Field descriptions for Styles

Field	Description
Weight Title	Allows you to define the following features in the Widget Title: • **Weight** - The **Weight** dropdown list lets you set the font thickness of the field label. • **Size** - Allows you to increase/decrease font **Size** of the field name.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
	• **Line height** - Allows you to increase/decrease the **Line height** of the field. This means the space between the field and the field border. • **Color** - Allows you to define the **Color** of the field label. • **Align** - Allows you to **Align** the field label to the left, center or right side of the field.
Grid Data Text	Allows you to define the following features for header text: • **Header Row Text Color** - Allows you to select the color for the text in the Header row from the color picker or the color palette. • **Header Row Text Size** - Allows you to set the size of the text in the Header row. • **Header text Alignment** - Allows you to align the text in the Header row to the left, center or right side. • **Data Text Color** - Allows you to select the color for the data text from the color picker or the color palette. • **Data Text Size** - Allows you to set the size of the data text in the Tabular. • **Data Text Alignment** - Allows you to align the data text to the left, center or right side.

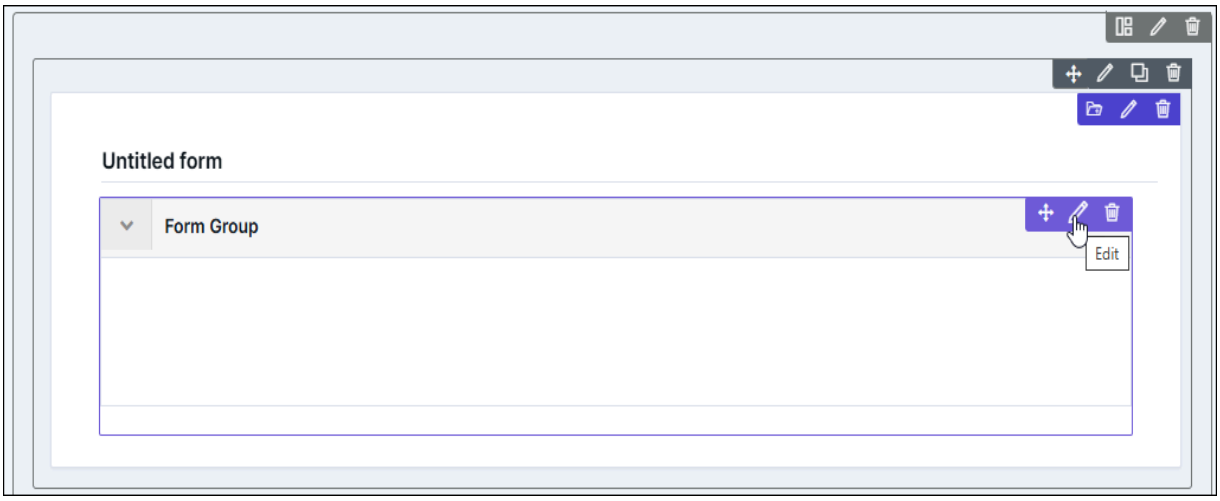
2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Form Field - Form Group

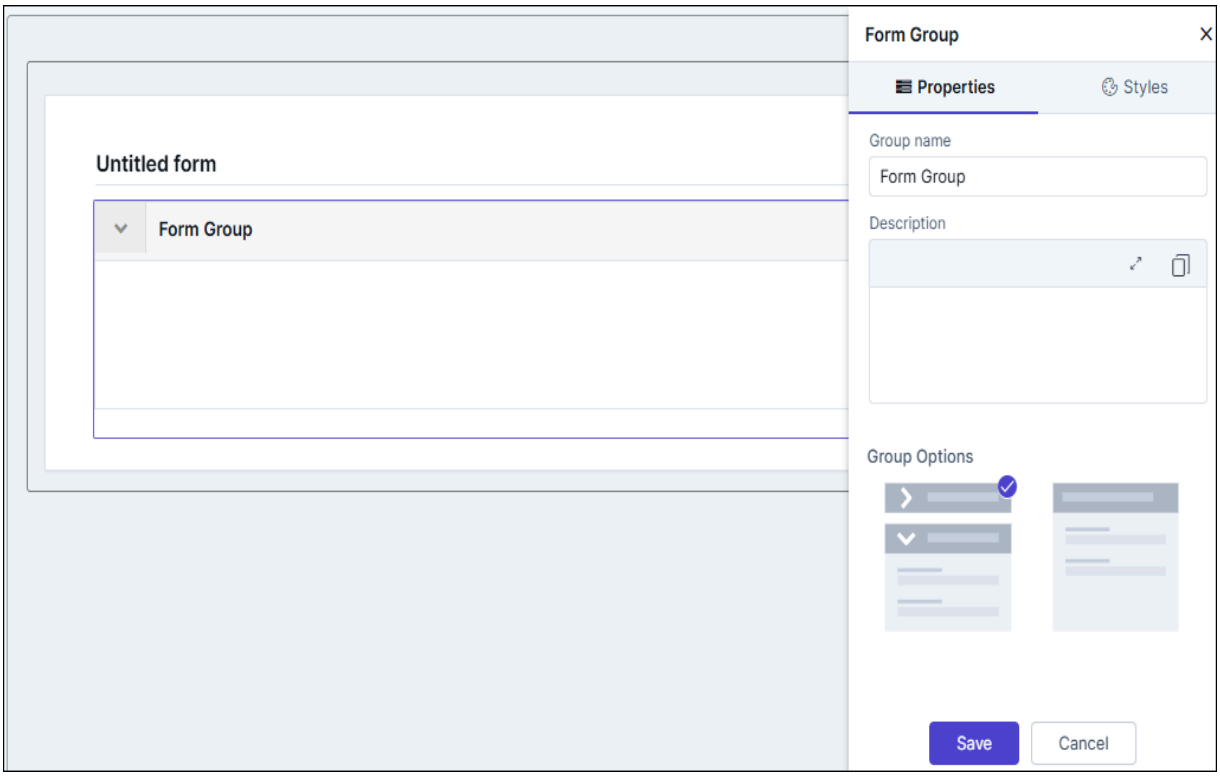
The Form Group component is used to group related form fields together, improving the form's organization and readability. It provides a logical division of related inputs.

1. Go to **(Menu)** > **Automation** > **Service Catalog** > **Design**.
2. To design a new form page, click the **+** (**Design**) icon.
3. Click the **+** (**New Page**) icon.
4. On the displayed new page, in the **Switch Layout** pop-up, select either **Single Page** or **Multi page** to proceed. (You may even close the pop-up without selecting any option. Single Page is the default layout applied.)
5. On the left click **Build**, and scroll down to view the **Form** components.
6. Select **Form Group** and drag it to the right on the new page.
   The Form Group component is added in the form, with icons to *Move*, *Edit*, and *Delete*.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



7. To modify the properties and styles click the Edit icon. Refer to the sections below.



Editing Form Group Properties

The edit panel opens on the right with **Properties** selected as the default tab.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Properties

Fields	Description
Group name	Enter a valid name for the component.
Description	Allows you to write a description for the component that will be displayed below the group name.
Group Options	Allows you to select from two options, drill-down and without. Choosing the drill-down option enables to expand/hide the grouped components.
Save / Cancel buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**.

Editing Form Group Styles

The edit panel opens on the right, click the **Styles** tab to configure the overall appearance.

Field description for Styles

Fields	Description
Group Title	
Weight	The Weight dropdown list lets you set the font thickness of the field label using one of the following values: Default, Light, Normal, Bold.
Size	Allows you to increase/decrease font size of the component.
Line height	Allows you to increase/decrease the line height of the field. This means the space between the field and the field border.
Color	Allows you to define the color of the component/label name.
Align	Allows you to align the field name to the left, center or right side of the form.
Group Outer Spacing	Allows you to increase/decrease the spacing between the outer border and the component.
Group Inner Container Spacing	Allows you to increase/decrease the spacing between the inner (added components) and the form group component.
Border Bottom	**Border Color** - Allows you to change the color of the border below the label name.

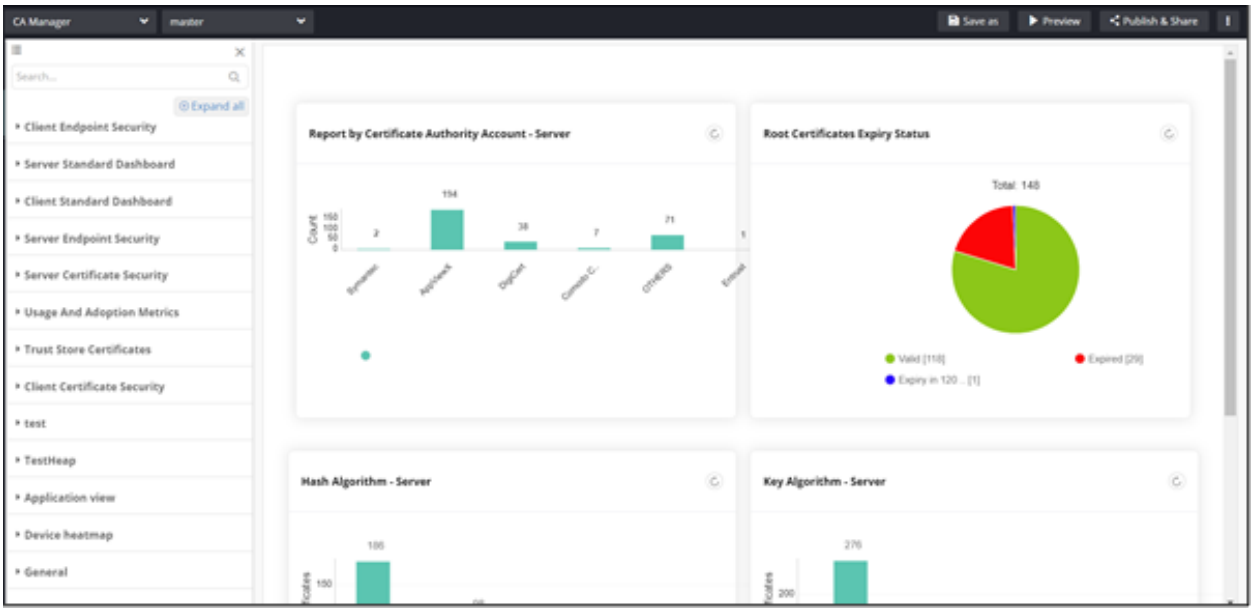
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field description for Styles (continued)

Fields	Description
	Border Width - Allows you to change the width of the border below the label name.
Save / Cancel / Restore buttons	To update the changes, click **Save**. To undo the changes, click **Cancel**. To restore the settings to default, click icon.

Reports

You can access specific reports based on user roles from the Service Catalog, enabling you to view all reports on a single page.

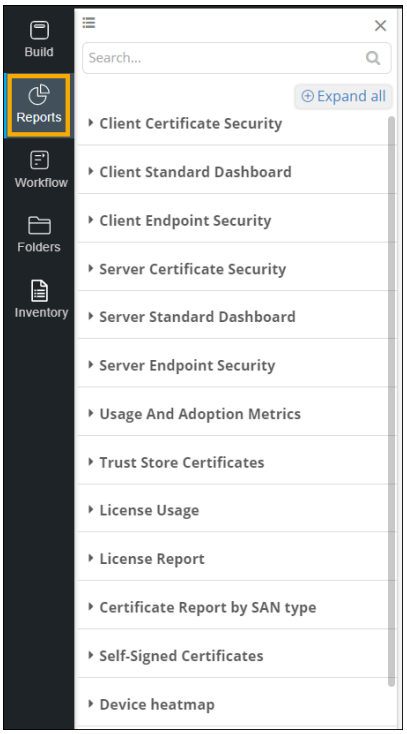


Note: For more information on **Reports**, refer to the Reporting User Guide.

To access Reports:

1. From the component panel on the left, click **Reports.**
   The **Reports** inventory is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

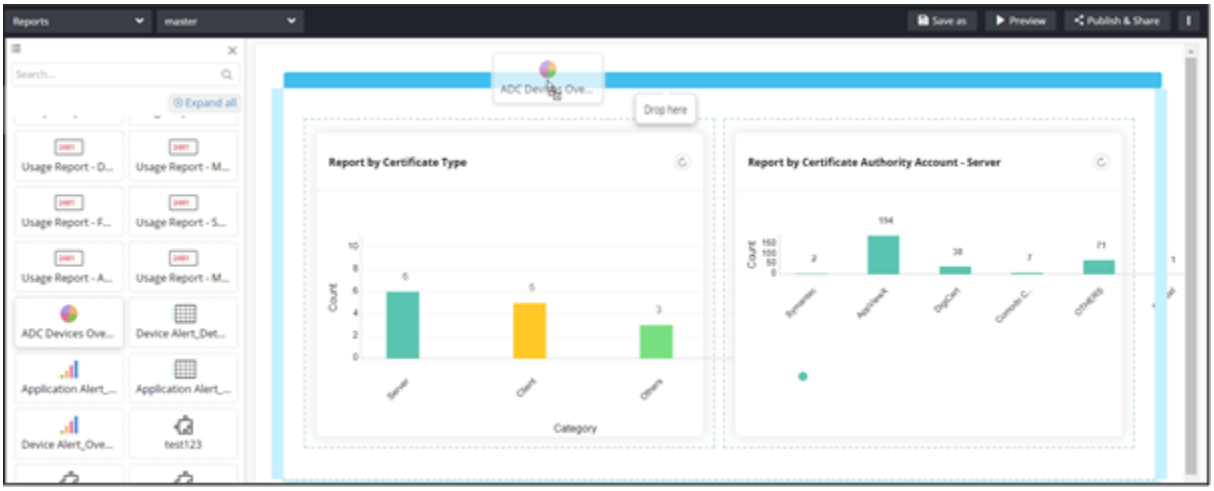


Options descriptions for Reports

Options	Description
View	Allows you to switch between list view and grid view when the Reports Inventory is expanded.
Search bar	Allow you to search for a specific report.
Collapse All/ Expand All	Allows you to expand/collapse the Reports inventory.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

2. To add a report to your page, drag and drop the report to any blue highlighted space in the build area.



- Configuring Reports - Properties
- Configuring Reports - Styles
- Pinning Application Widgets
- Pinning Device heatmap Widget

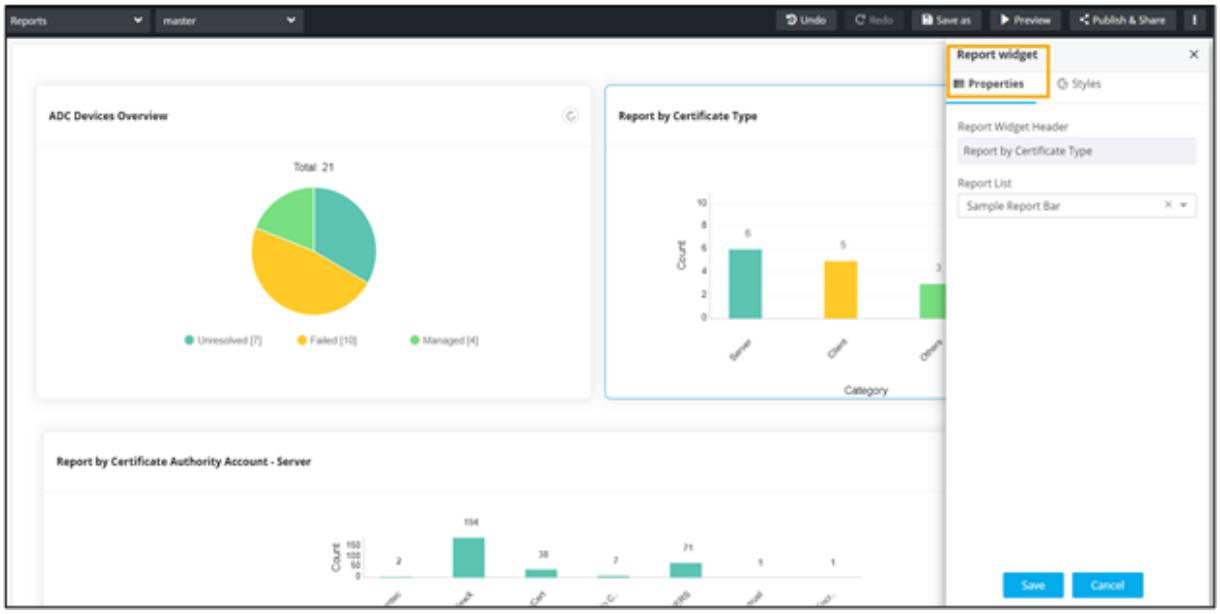
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Configuring Reports - Properties

1. To configure/edit the **Reports** component, hover your mouse over the widget and click **(Edit)** icon.

The **Report widget** window opens with the **Properties** tab open by default.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



2. In the **Report widget** window, under **Properties**, enter or select the required field information.

Field descriptions for Properties

Field	Description
Report Widget Header	Allows you to provide a name for the Report Widget.
Report List	Allows you to select reports from the options available in the dropdown. **Note:** Selecting a report from this dropdown will replace the existing report on the page.
Select icon	Allows you to select an icon to be displayed next to the widget name.

Chart Settings	Allows you to choose a template, fit, and alignment for the widget as follows:
	• **Template**
	  • Classic
	  • Standard
	  • Minimal
	• **Fit**
	  • Auto
	  • Relaxed

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
	• **Alignment** - • Left • Center

3. Click **Save**.

Configuring Reports - Styles

1. For more options to customize the **Report widget**, in the **Report widget** window, click **Styles**.

Field descriptions for Styles

Field	Description
Sizing	Allows you to increase/decrease the **Size** of the Report widget.
Spacing	Allows you to set the **Padding** and **Margin** of the Report widget header.
Widget Background	The following options are available here: • **Add Background Pattern**: To add a background pattern to the widget, select the **Add Background Pattern** checkbox and select a pattern from the available options. • **Widget Background Color**: To add a background color to the widget, deselect the **Add Background Pattern** checkbox and select the color from the color palette. • **Widget Border Color**: Allows you to change the color of the widget border.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Widget Header	The following options are available here: • **Header Text Color**: Allows you to change the color of the header text. • **Header Text Size**: Allows you to select the size of the header text. • **Header Text Underline**: To underline the header text, select the Header Text Underline checkbox.
Icon Style	The following options are available here: • **Icon Color** - Select the Icon Color from the available options. • **Icon Background** - Select the Icon Background from the available options. • **Icon Background Color** - Select the Icon Background Color from the available options.

2. To save your settings, click **Save**.
3. To restore the settings under **Styles** to default, click **(Restore)** icon.

Pinning Application Widgets

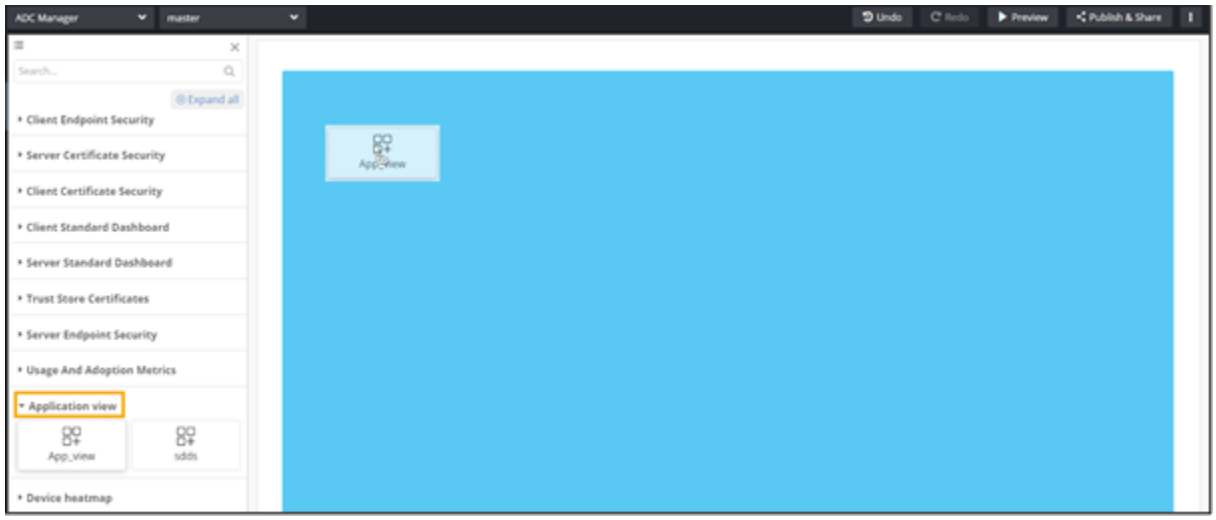
Through the Service Catalog, you can pin Application view and Device HeatMap widgets to your page. To be able to pin these widgets to your page, they must be configured on the ADC dashboard first.

The Application View widget displays all the configured devices in your database. Through the widget you can also perform actions such as enabling/disabling a device, view logs and alerts, backup and restore and so on.

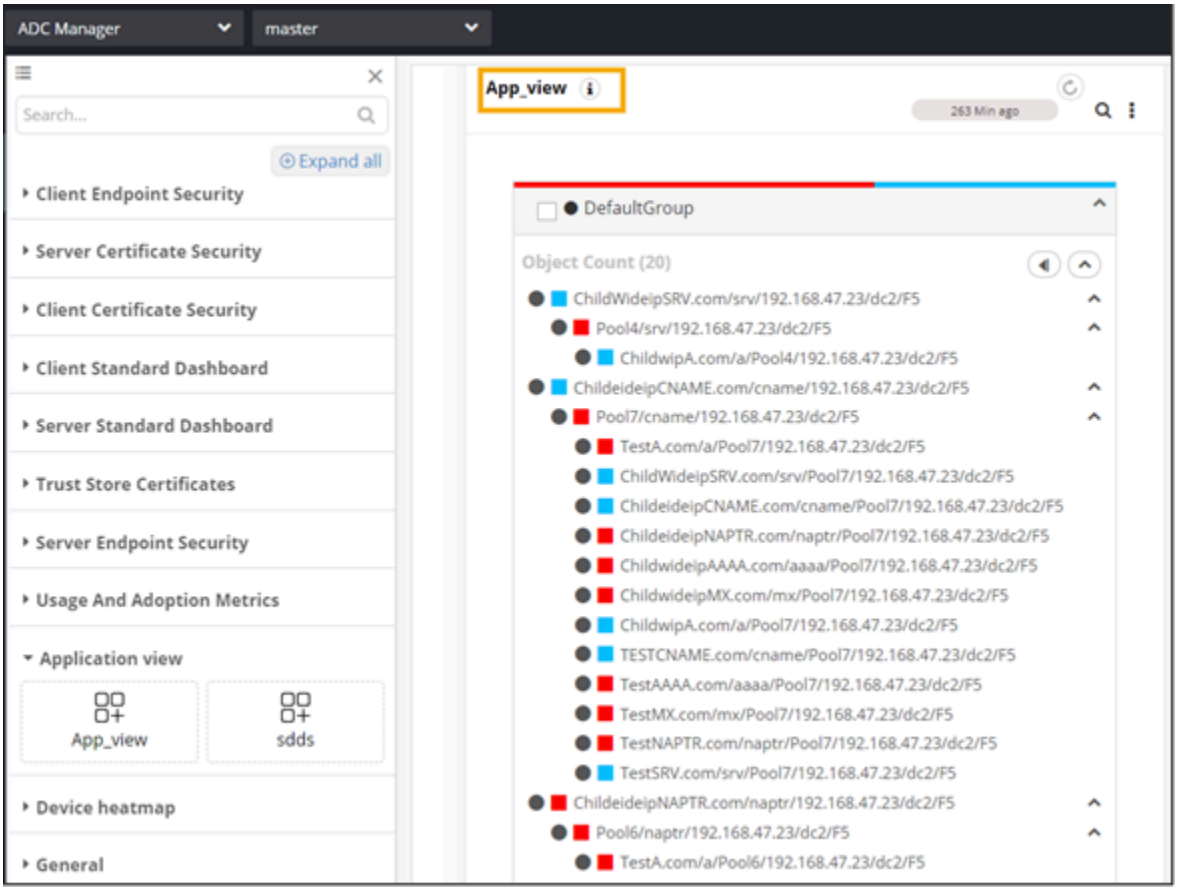
To pin the Application view widget to your page:

1. From the component panel on the left, click **Reports**.
2. Under **Application view**, drag and drop the widget that you want to pin to the page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The **App_view** widget displays all the devices configured in the database.

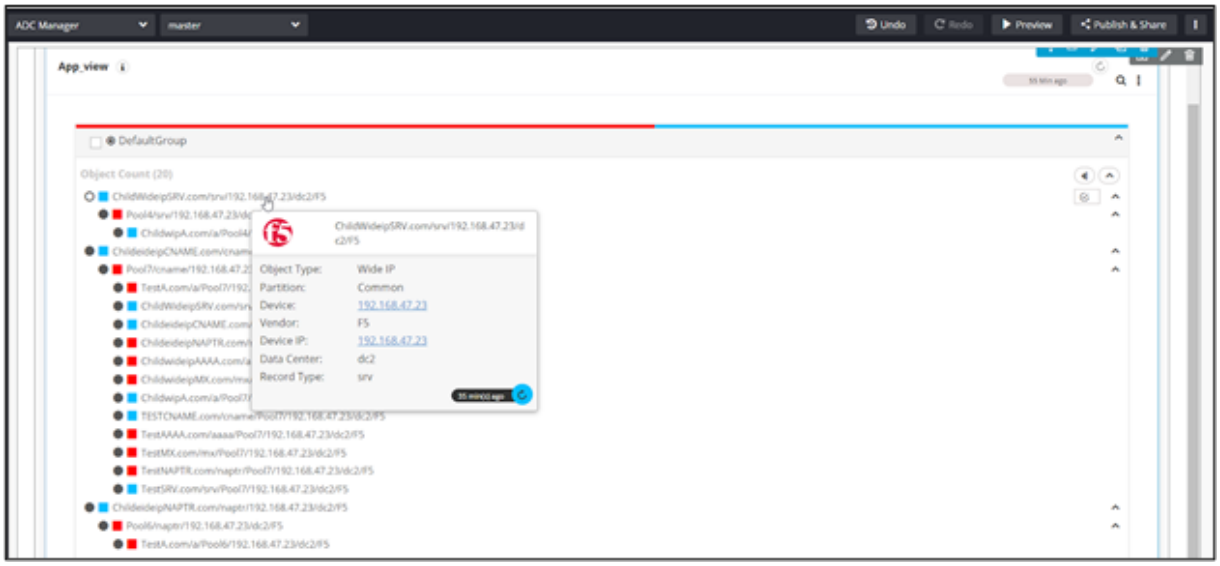


3. To modify the widget, hover your mouse over the top right corner of the widget and click **(Edit)** icon.

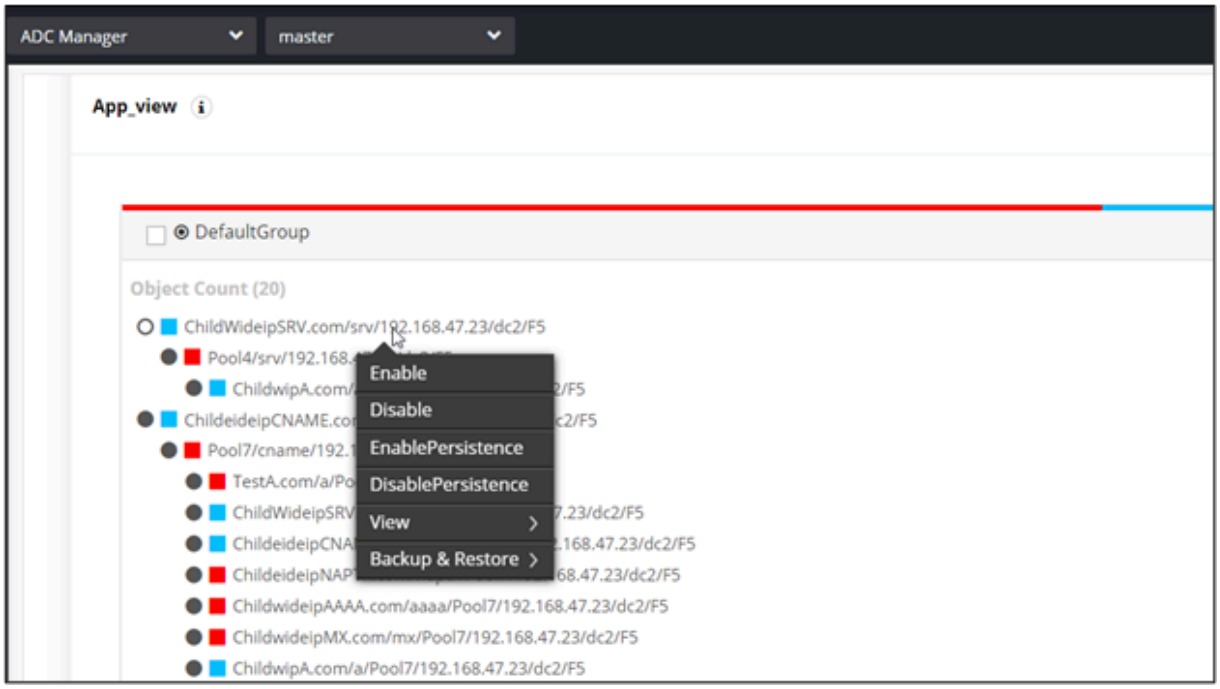
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Note: For more information on configuring the Report widget, click here.

4. To view details about any device, hover your mouse over the device name.

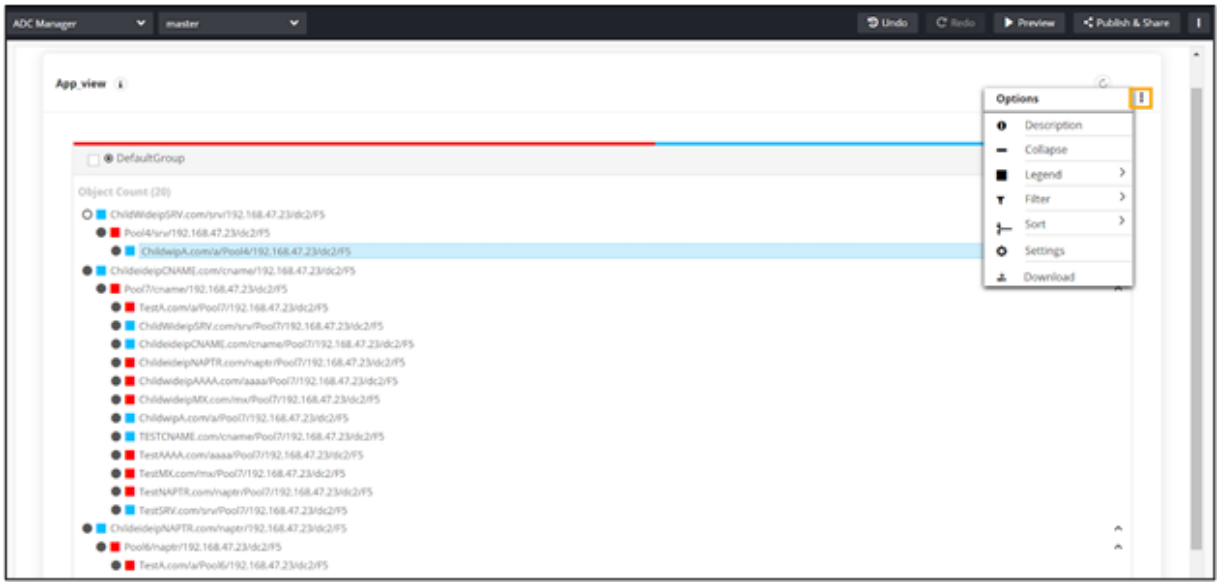


5. To see the list of actions that can be performed on these devices, right-click on a device name.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

6. To see the list of available options for the App_view widget, from the top right corner of the widget, click ⋮.

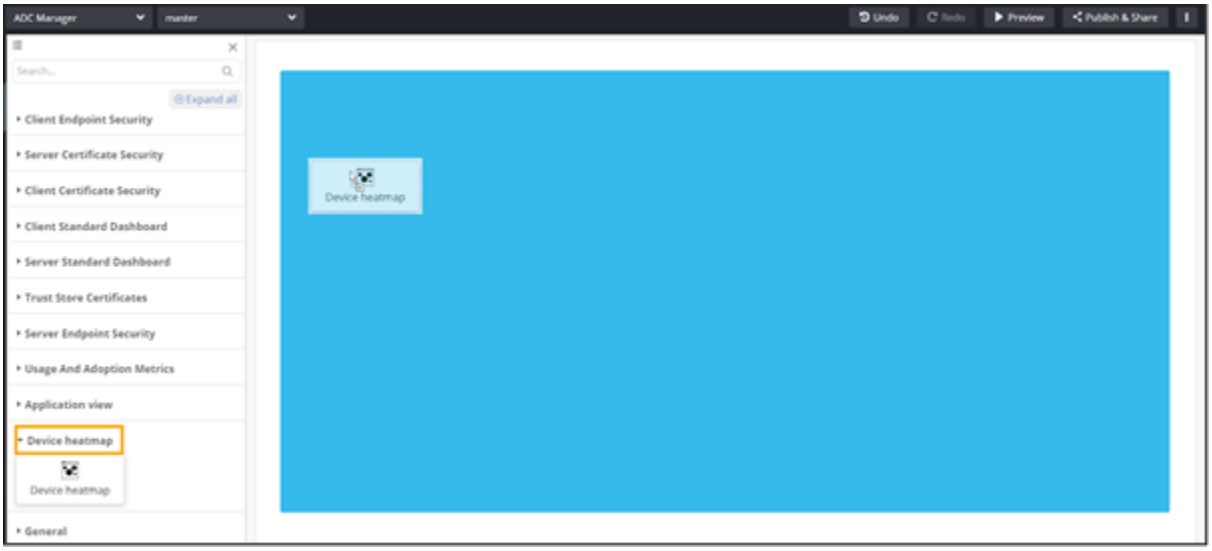


Pinning Device heatmap Widget

The **Device heatmap** widget shows a color-coded status of your configured devices.

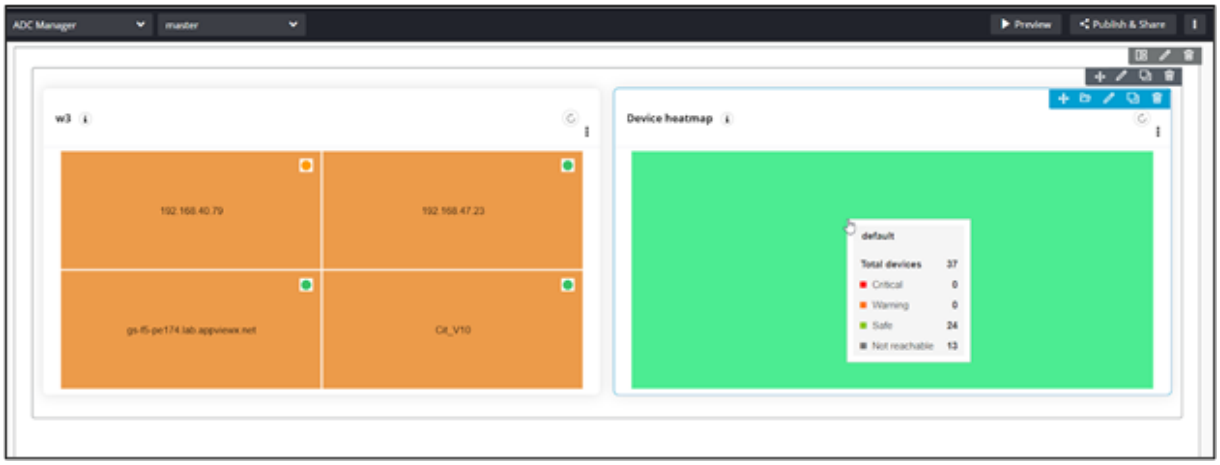
To pin the **Device heatmap** widget to your page:

1. From the component panel on the left, click **Reports**.
2. Under **Device heatmap**, drag and drop the widget that you want to pin to the page.



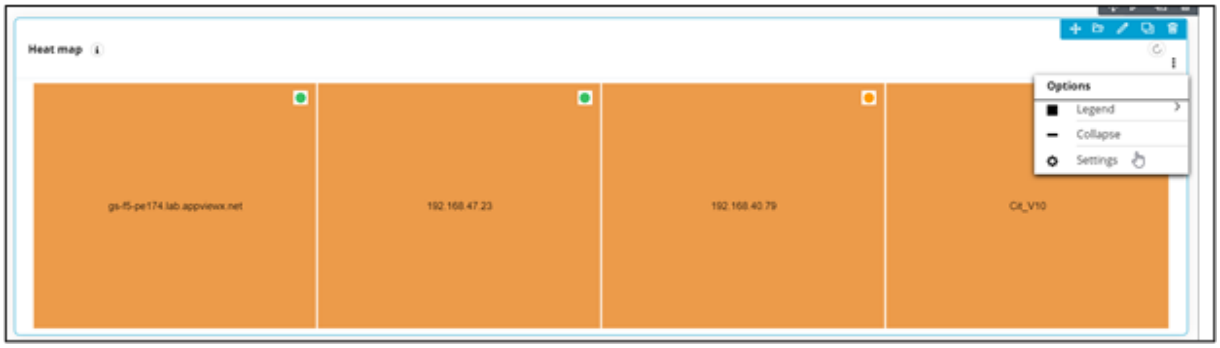
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

3. To modify the widget, hover your mouse over the top right corner of the widget and click **(Edit)** icon.



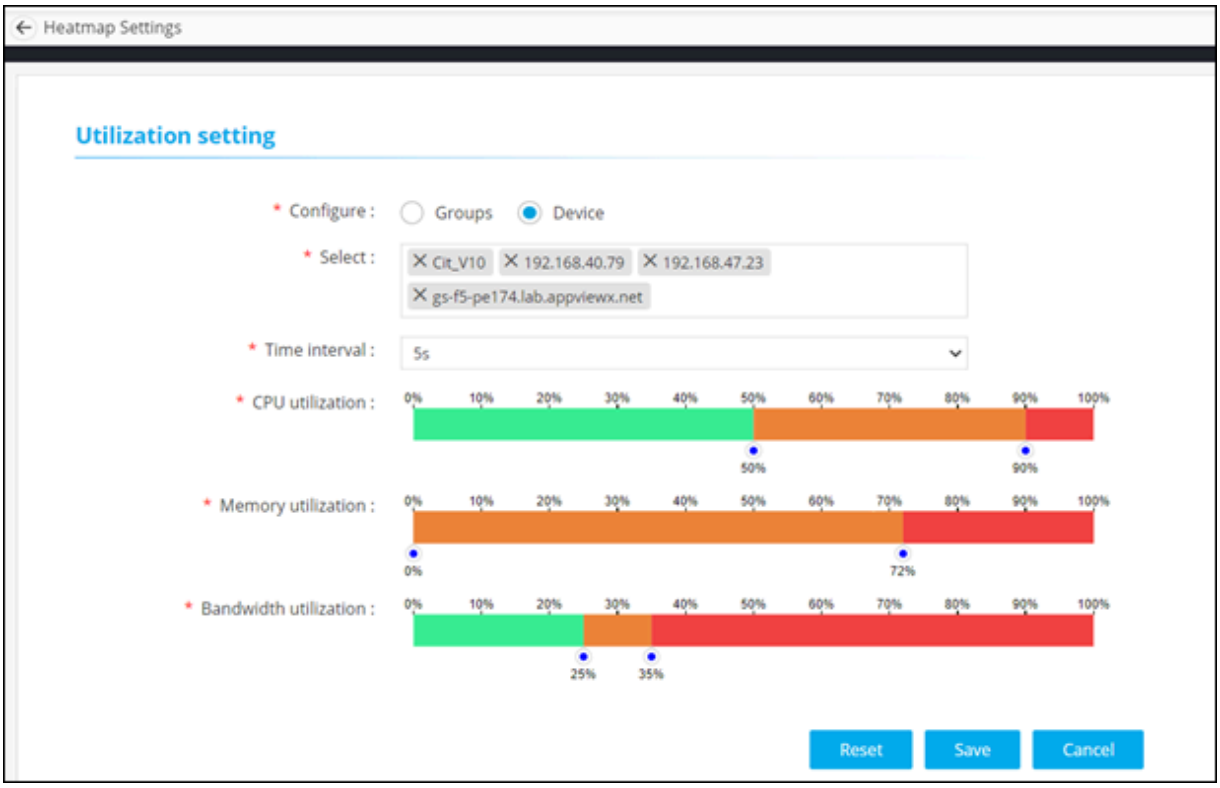
Note: For more information on configuring the Report widget, click here.

4. To modify the widget settings, from the top right corner of the widget, click .
5. From the options, select **Settings**.

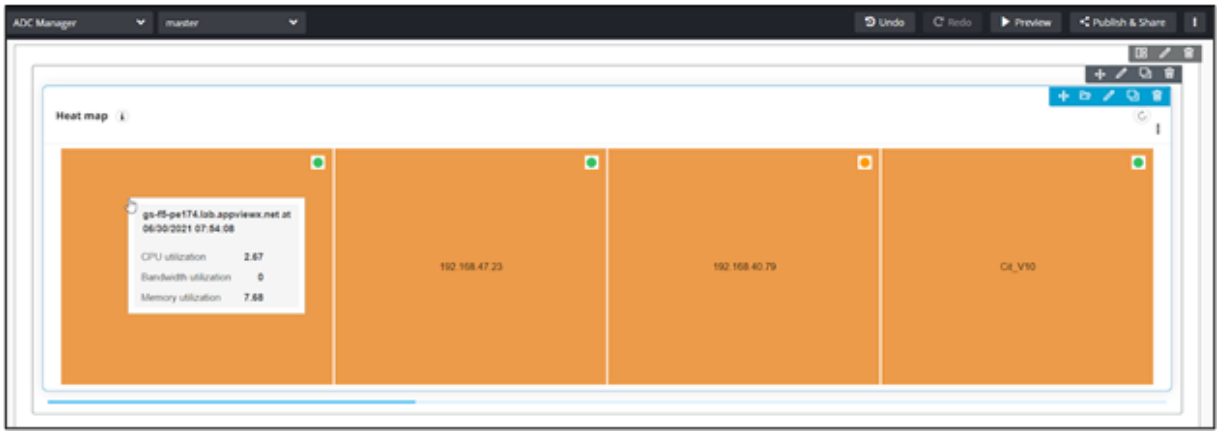


Note: You can configure devices and groups and other settings here.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



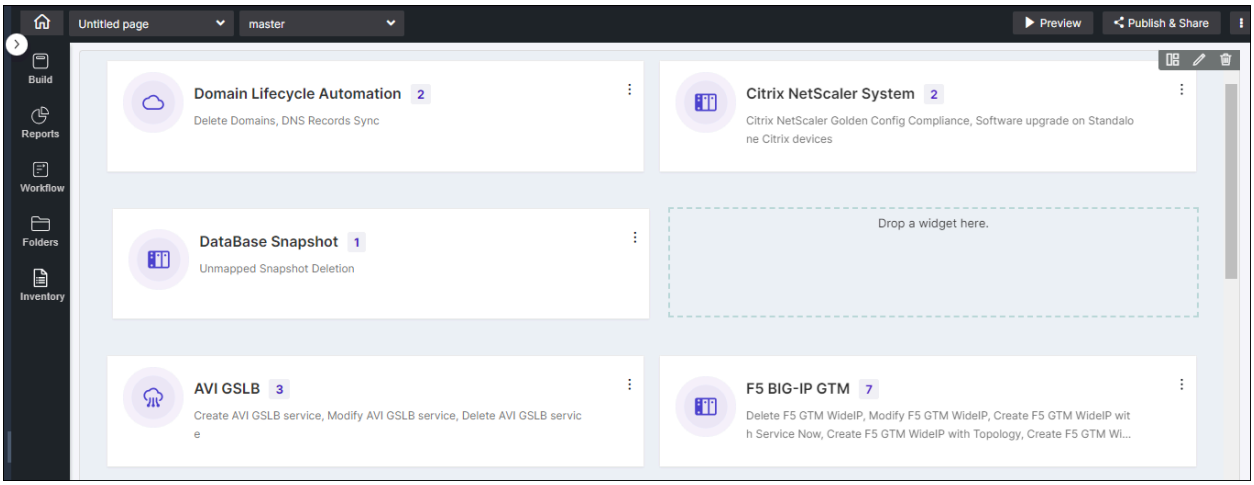
6. For details about a particular device on the heat map, hover your mouse over the device.



Workflow

The Workflow component allows you to use the service catalog to execute automation workflows by persona. You can logically select and group workflows such as CLM Automation, Application Delivery and so on.

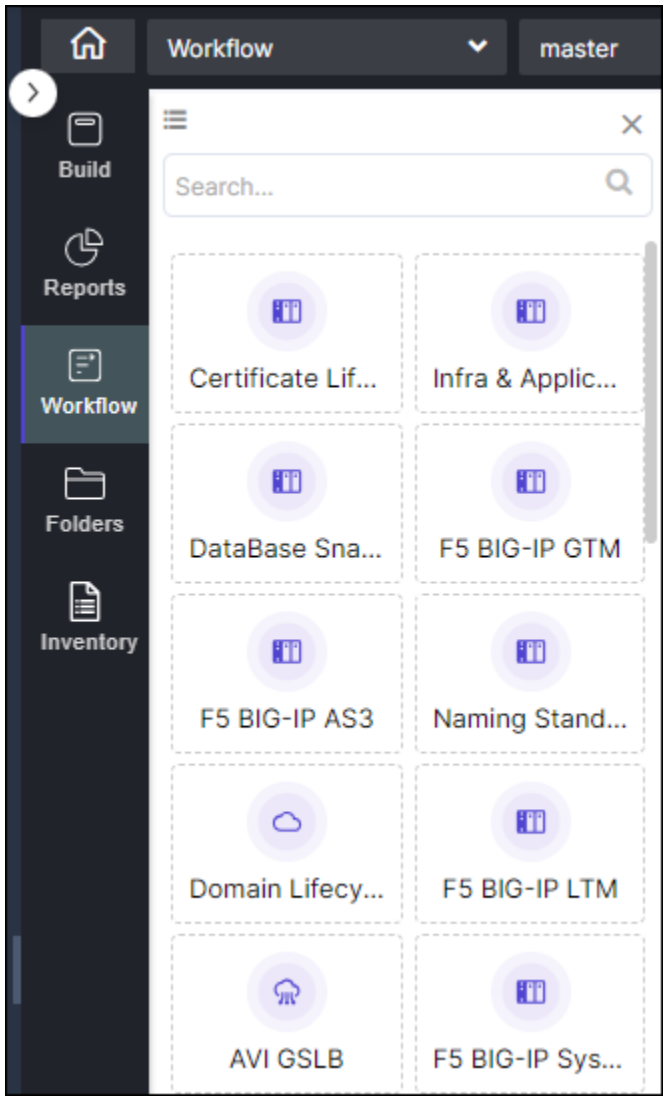
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



To access workflow catalogs:

1. From the component panel on the left, click **Workflow**.
   A list of available workflow catalogs is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

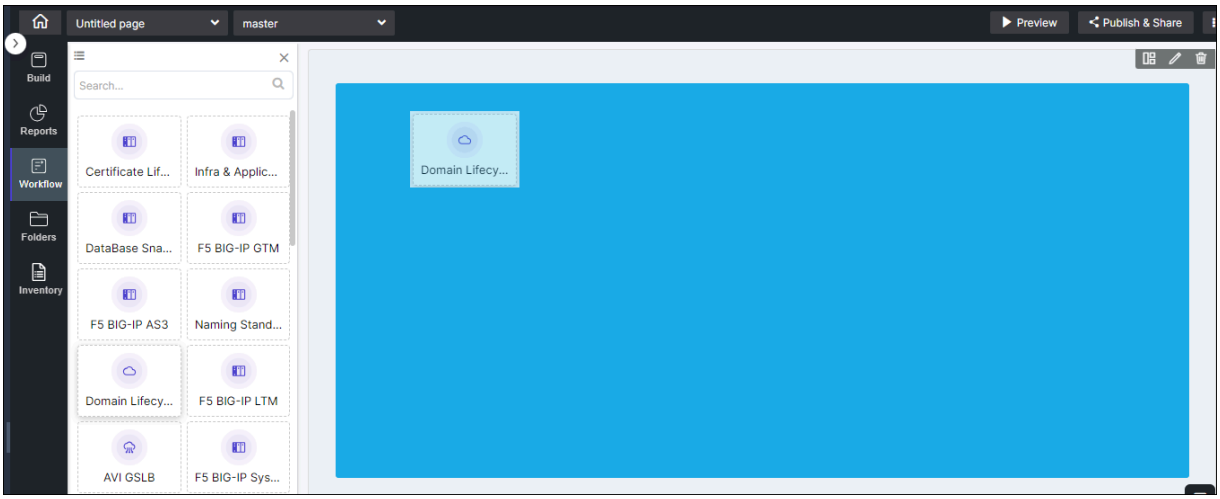


This table describes the options present in the **Workflow** section:

Field	Description
	Allows you to switch between list view and grid view of the Workflow catalogs.
Search bar	Allows you to search for a specific workflow catalog.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

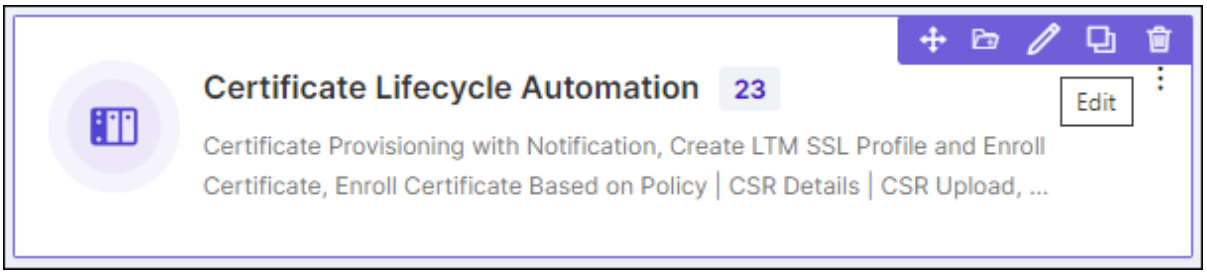
2. To add a workflow catalog to your page, drag and drop the selected workflow to any blue highlighted space in the build area.



- Configuring Workflow - Properties
- Configuring Workflow - Styles

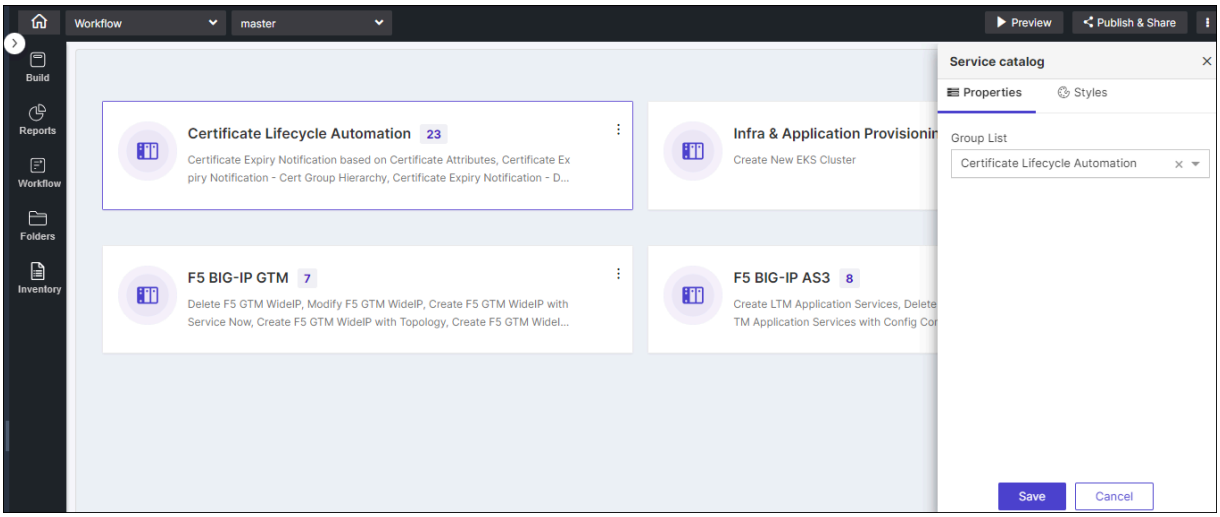
Configuring Workflow - Properties

1. To configure/edit the **Workflow** component, hover your mouse over the widget and click **(Edit)** icon.



The **Service Catalog** window opens with the **Properties** tab open by default.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



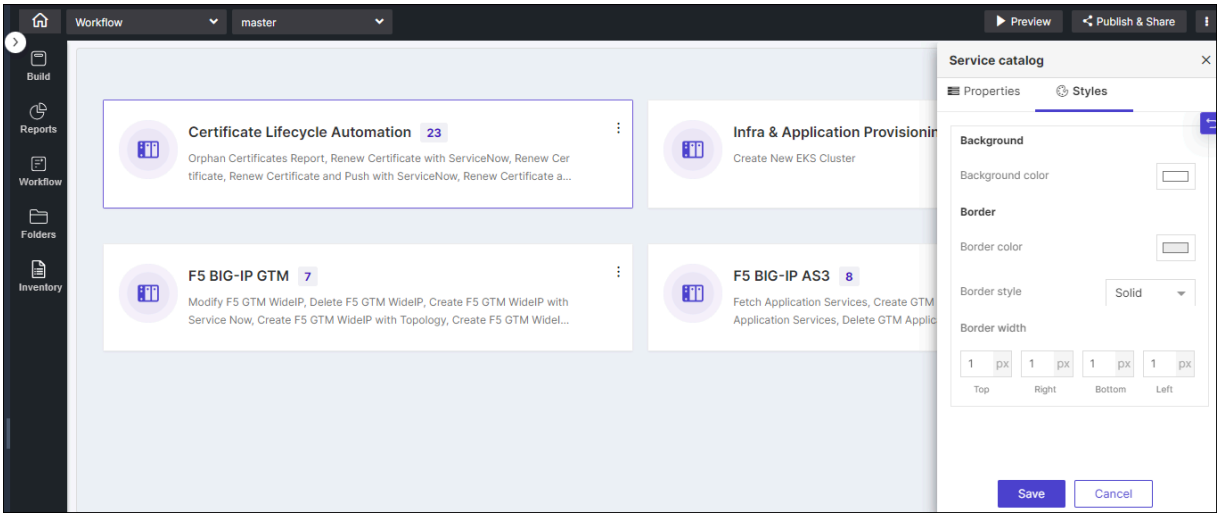
2. In the **Service catalog** window, under **Properties**, select a workflow catalog from the **Group List** dropdown.

> **Note:** The selected catalog will replace the catalog existing on the page.

3. Click **Save**.

Configuring Workflow - Styles

1. For more options to customize the **Workflow** widget, in the **Service catalog** window, click **Styles**.



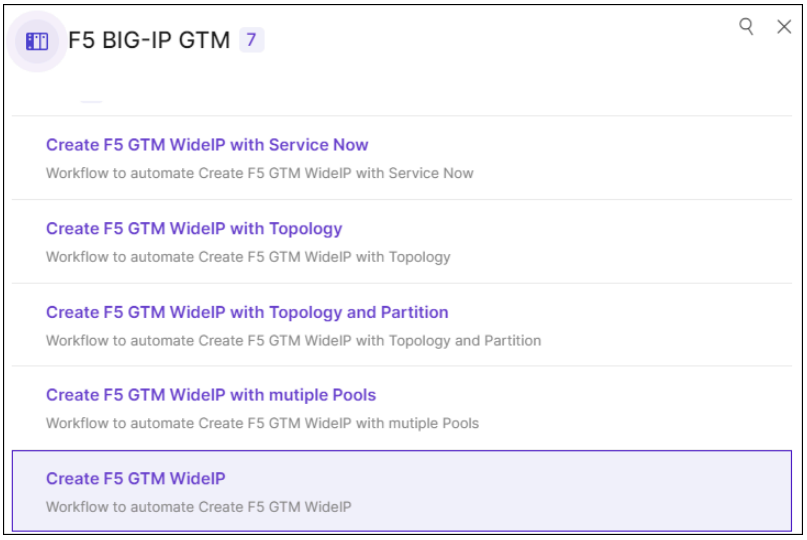
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Styles

Option	Description
Background	Allows you to set the background color of the workflow widget.
Border	Allows you to define the • **Border color** - Set the color of the Workflow widget's border. • **Border style** - Set the border style for the Workflow widget from the options available in the dropdown. • **Border width** - Set the thickness of the Workflow widget's border.

2. To trigger workflows from your page, click **Preview**.

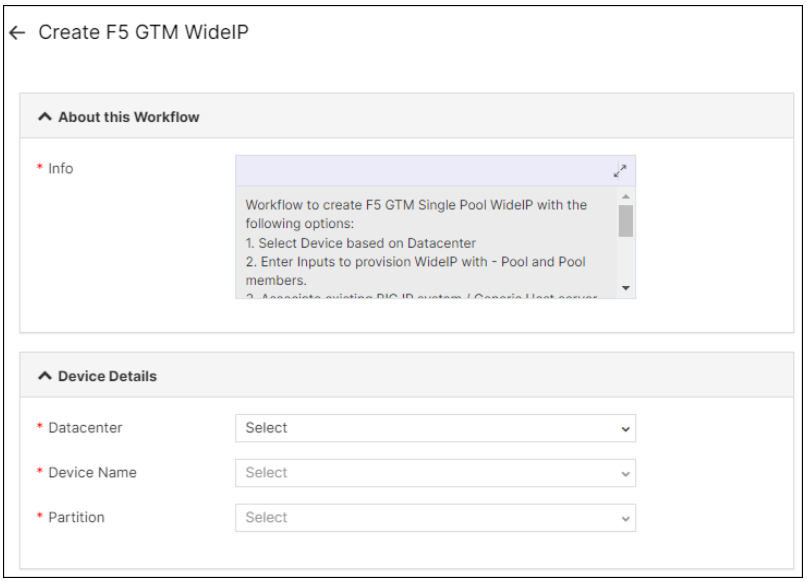
   A preview of the page opens in a new tab.



3. To trigger a workflow locate the workflow in the catalog and click the catalog name.

   The selected workflow is triggered.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



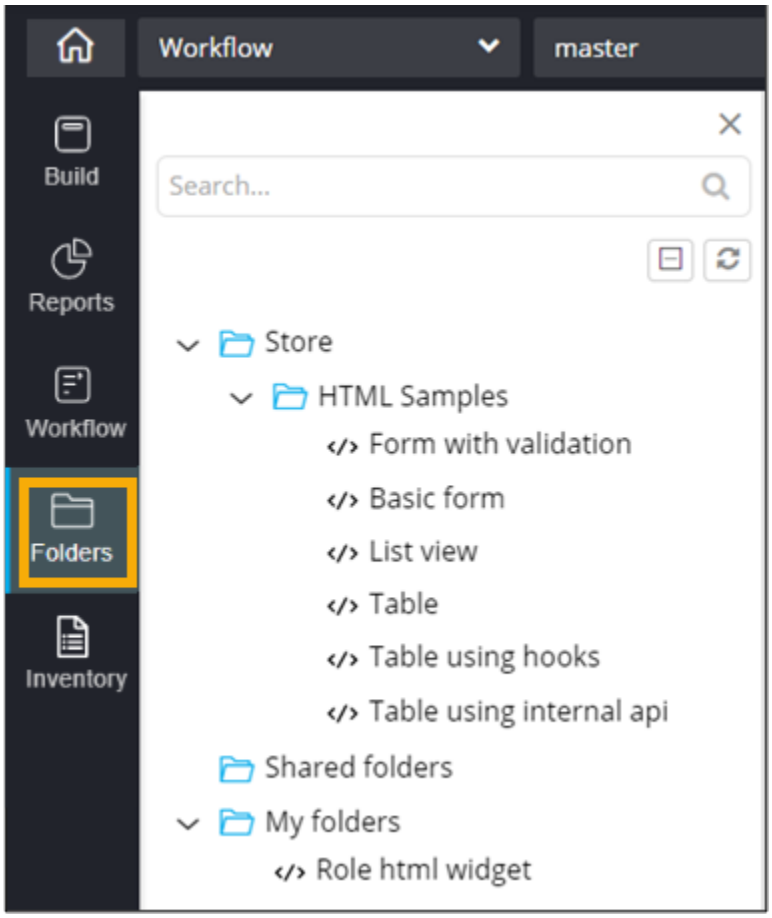
Note: For more information on workflows, refer to the Automation User Guide.

Folders

You can reuse widgets by saving them in folders. The Folders section also provides sample HTML scripts. Folders enable reusability of elements across multiple users thereby saving time.

The **Folders** can be accessed from the component panel on the left.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



This table describes the options present in this section:

Options descriptions

Option	Description
Search bar	Allows you to search for a specific folder, widget or sample HTML script by typing keyword(s) in the search field.
/	Allows you to toggle between expanded and collapsed views of the folders.
	Refresh the folders inventory.

- Saving a Widget in Folders
- Deleting Folders

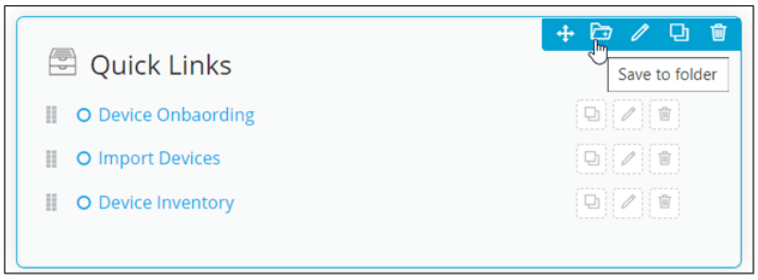
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Saving a Widget in Folders

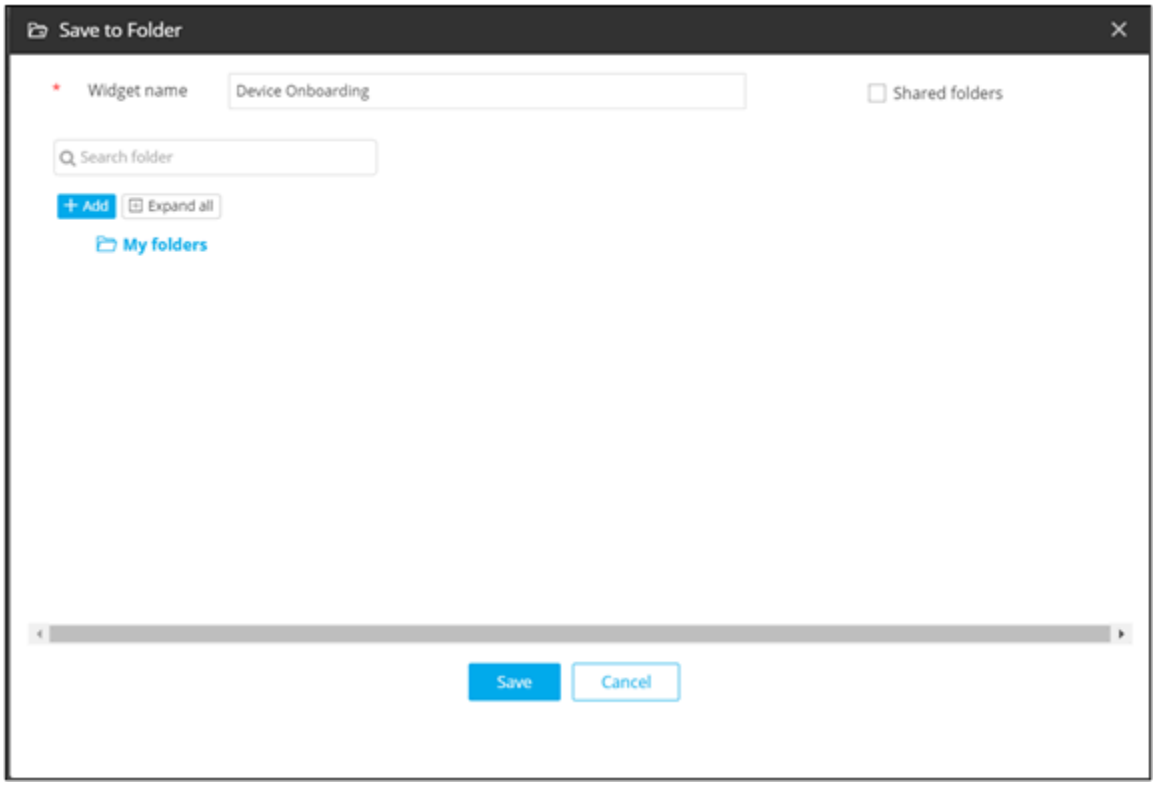
You can save one or more page/widget elements into a custom folder.

To save a widget to a folder:

1. From the top right corner of the widget to be saved, click **(Folder)** icon.



2. In the **Save to Folder** window, enter or select the field information.



Field descriptions for Save to Folder

Field	Description
***Widget name**	Enter a suitable widget name.
Search bar	Allows users to search for a particular folder.

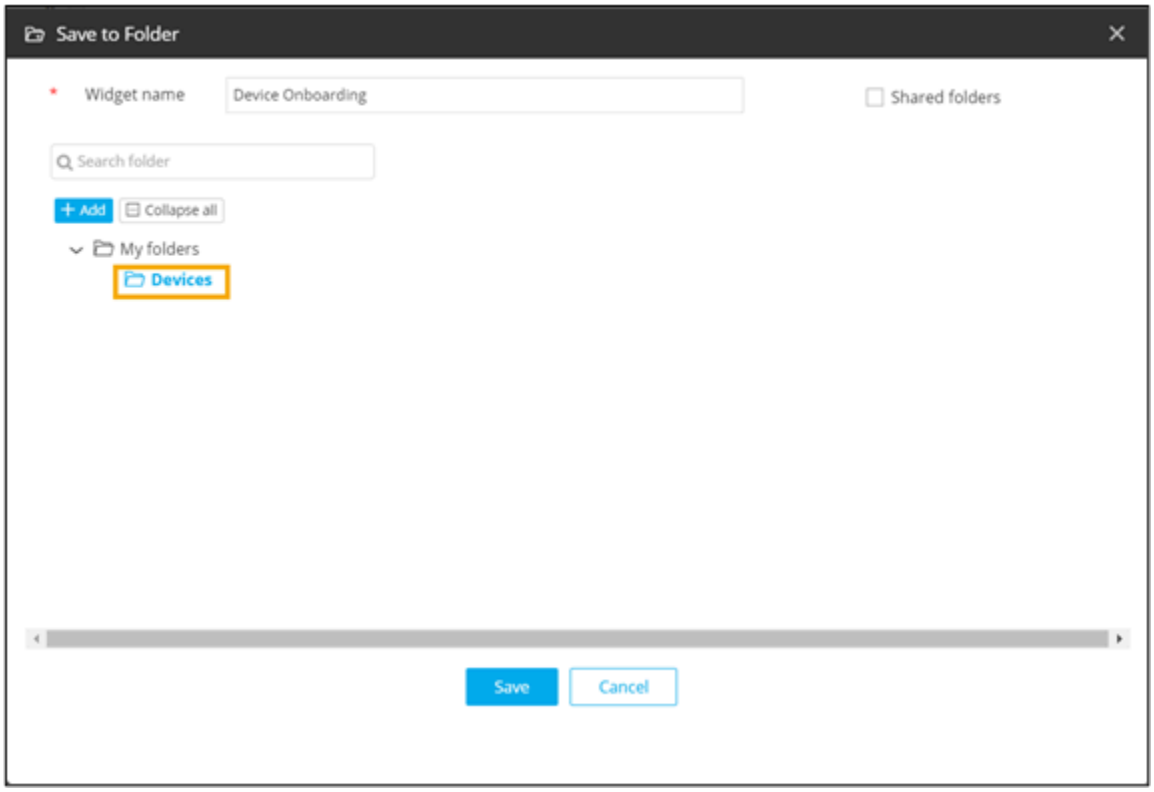
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field	Description
Add	Allows users to add new folders and sub-folders.
Expand All	Allows users to expand and collapse folders.
My folders	Shows sub-folders within My folders.
Shared folders	Allows users to save a widget in a shared folder.
**: Mandatory fields*	

Saving a widget in My Folders

To save a widget to be reused only by the user who created the widget:

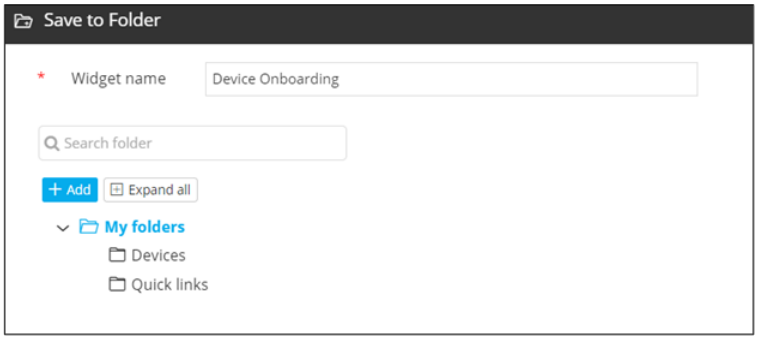
1. In the **Save to Folder** window, under **My folders**, click on the folder in which you want the widget to be saved.



2. To create a new folder for saving the widget, click **Add**.
3. Enter the folder name.

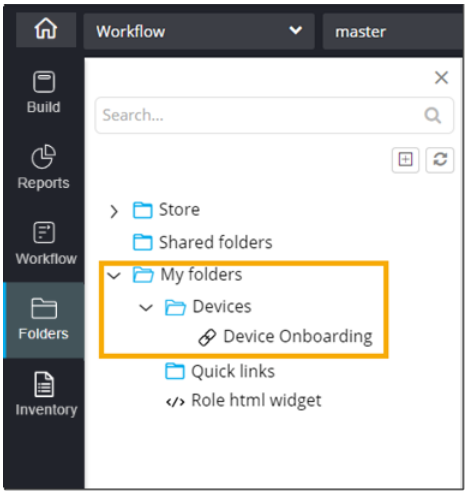
   A new folder is created.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. Select the folder in which you want to save the widget.
5. Click **Save**.

   The widget is saved in the selected folder.

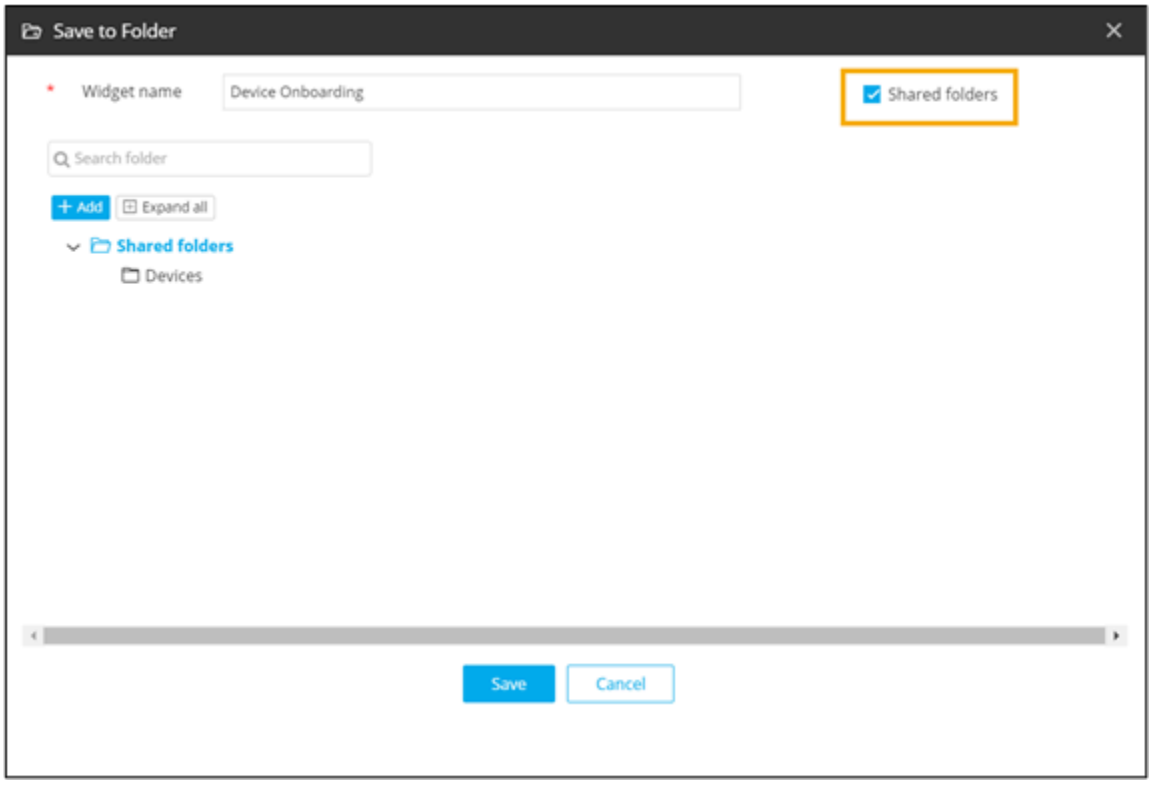


Saving a widget in Shared Folders

To save a widget that can be shared with other users:

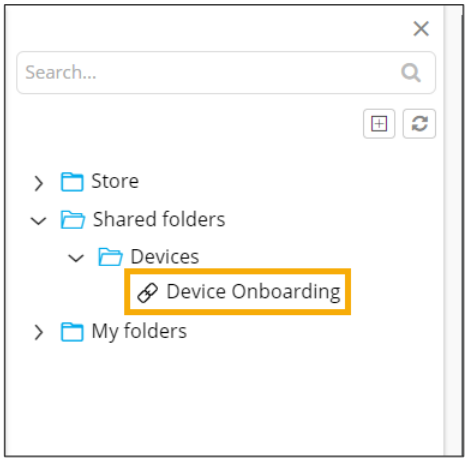
Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. In the **Save to Folder** window, select the **Shared Folders** checkbox.



2. Click on the shared folder where you want the widget to be saved. For example, Devices.
3. To create a new folder under the Shared folders, click **Add**.
4. Select the required folder and click **Save**.

   The widget is saved in the selected shared folder.

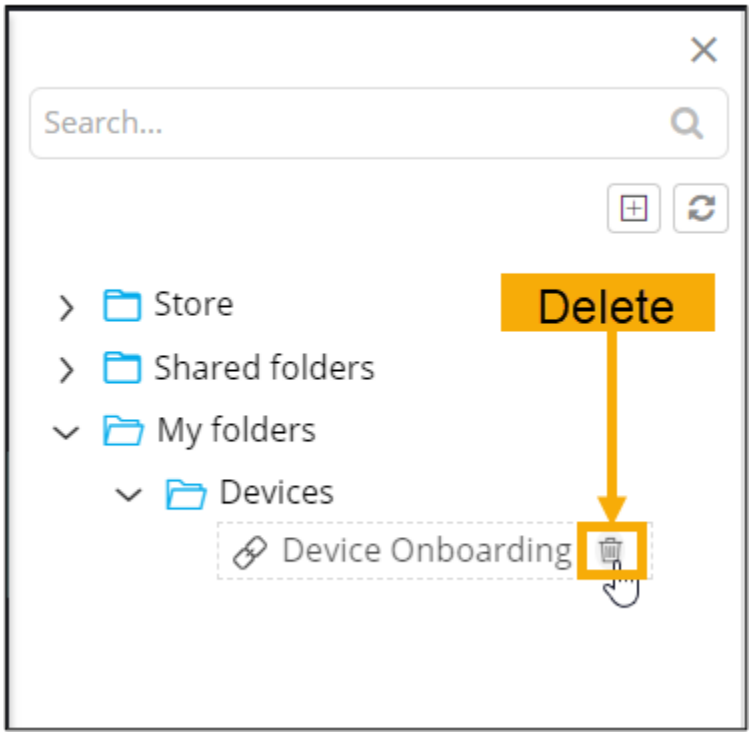


Deleting Folders

To delete a folder from My Folder or Shared Folders:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. From the component panel on the left, click **Folders**.
2. Click **(Delete)** icon next to the folder name.



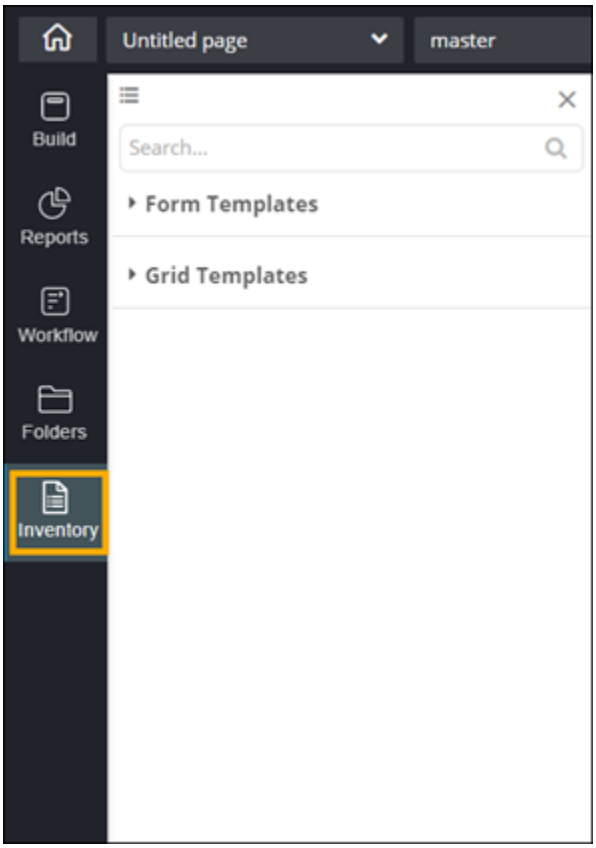
3. In the **Confirmation** window, click **Yes**.
   The folder is deleted.

Inventory

Service Catalog has an inventory of pre-built form and grid templates that can be used out of the box for quick self-servicing or customized as per requirement. 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.

The **Inventory** can be accessed from the component panel on the left.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Options available with descriptions in Inventory

Option	Description
	Allows you to switch between a list view and grid view of the forms.
Search	Allows you to search for a specific OOB or custom form.
Form Templates	This folder contains a set of prebuilt out of the box (OOB) forms that you can add to your page for
Grid Templates	This folder contains a set of prebuilt out of the box (OOB) grid templates that you can add to your p servicing.

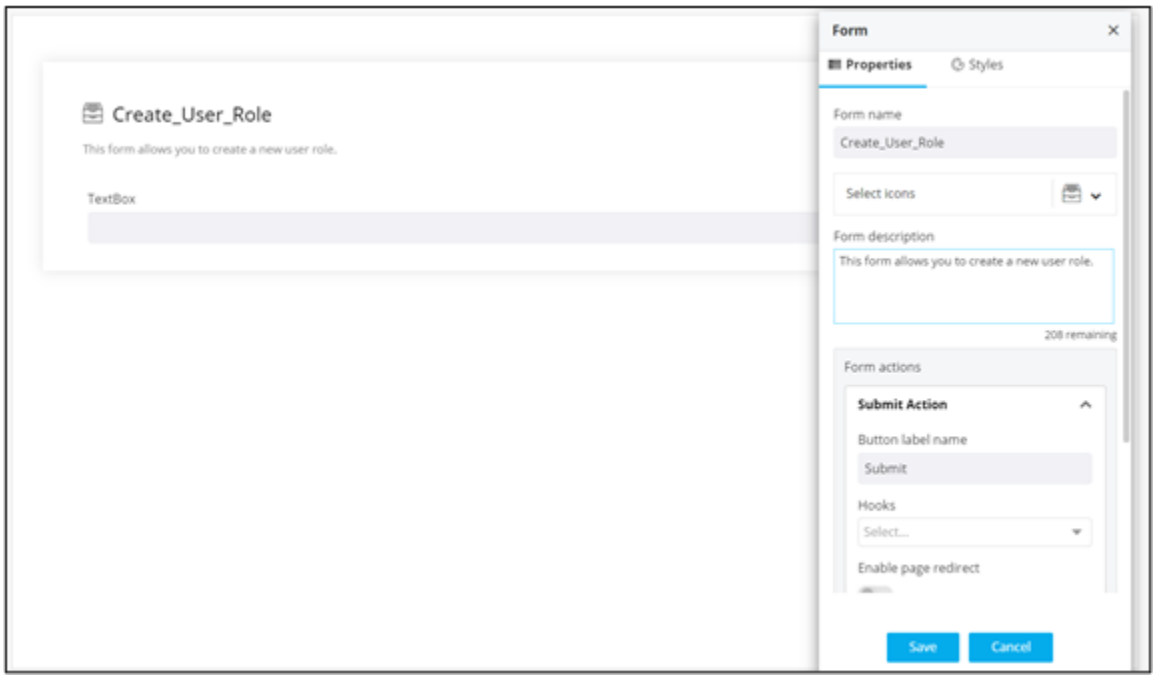
- Getting started with designing a Custom Form
- Using Form Templates
- Enabling Auto trigger

Getting started with designing a Custom Form

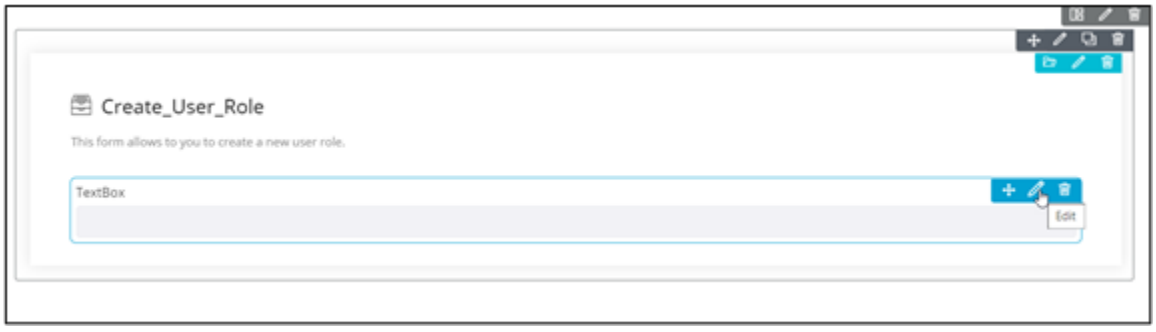
This example will help you understand how to design a basic form for creating a new role.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Design a new page.
2. Under **Build** > **Form**, drag and drop the **Text Box** form field to the design space.
3. To change the form name, hover your mouse over the form widget and from the top right corner, click **(Edit)** icon.
4. In the **Form** pop-up window, under **Properties**, change the **Form name** to **Create_User_Role**.
5. In the **Form** pop-up window, under **Properties**, add the **Form description**.



6. Click **Save**.
7. To rename the **Text Box** field, hover your mouse over the field and click **(Edit)** icon.



8. In the **Text Box** pop-up window, under **Properties**, enter or select the following field information:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Properties

Field	Description
***Label name**	Enter a valid **Label name** for the form field. For example: **Role name**.
***Field ID**	Enter a unique Field ID.
Mandatory	Turn on the toggle to make this field mandatory.
**: Mandatory fields*	

9. Click **Save**.
10. To add another Text Box field to the same form, drag and drop the **Text Box** widget within the existing form space.

11. To configure this **Text Box** field, hover your mouse over the Text Box field and click **(Edit)** icon.
12. In the **Text Box** pop-up window, under **Properties**, enter or select the following field information:

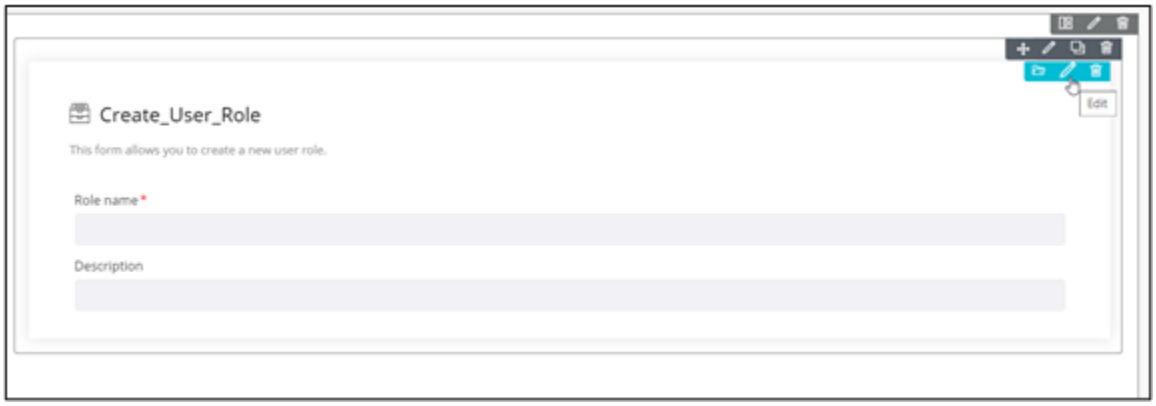
Field descriptions for Properties

Field	Description
***Label name**	Enter a valid label name for this field. For example, Role name.
***Field ID**	Enter a unique Field ID.
**: Mandatory fields*	

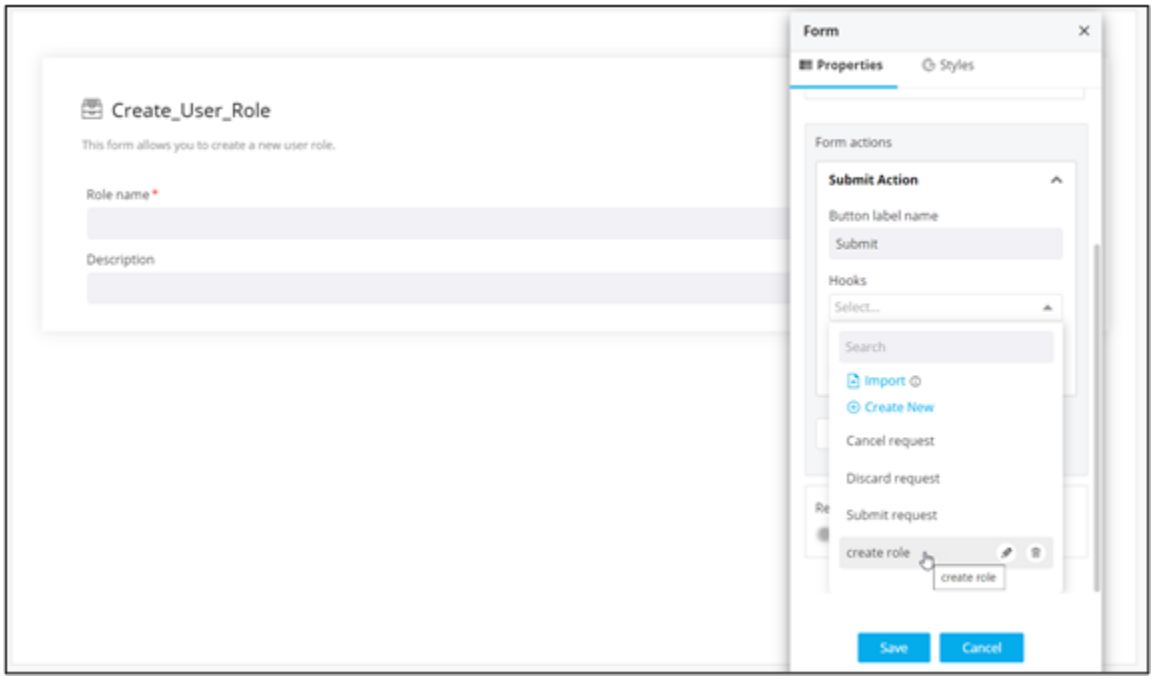
13. Click **Save**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

14. To add a Submit button to the form, hover your mouse over the form and from the top right corner of the form, click **(Edit)** icon.



15. In the **Form** pop-up window, under **Properties,** scroll down to **Form actions**.
16. Under **Form actions**, select, import, or create a new hook for the Submit form action.



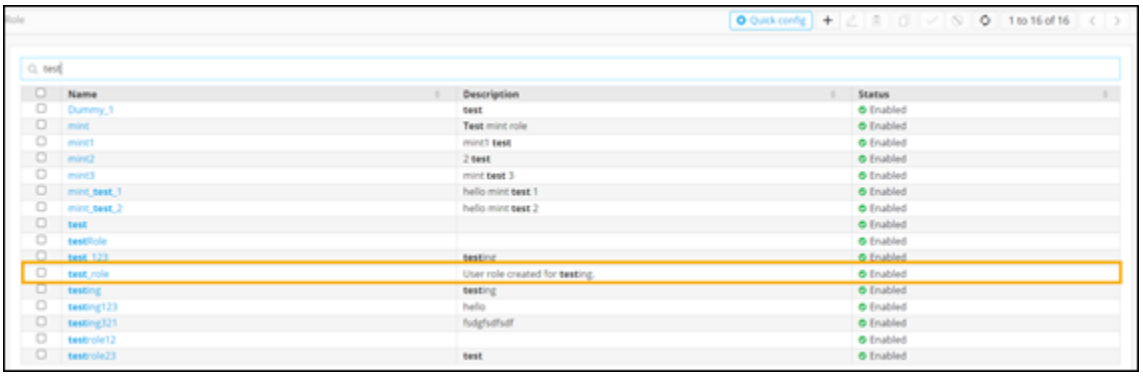
17. Click **Save**.
18. To use this form to create a new user role, click **Preview**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

The Preview page opens in a new tab.

19. Enter the **Role name** and **Description** for the new user.
20. Click **Submit**.

    The new user role is created and added to the list of Roles.



- Redirecting to a Page after Submit

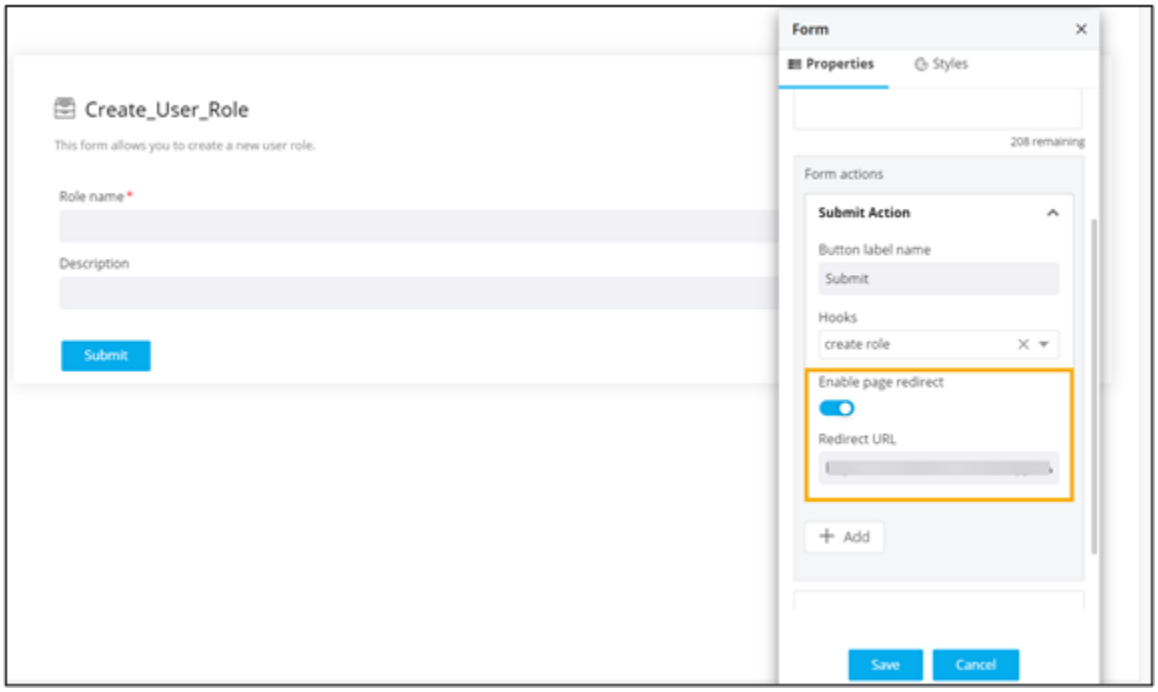
Redirecting to a Page after Submit

When designing a form, there is a provision to define the form action to redirect to another page in a new tab.

To add redirect action to the Submit form:

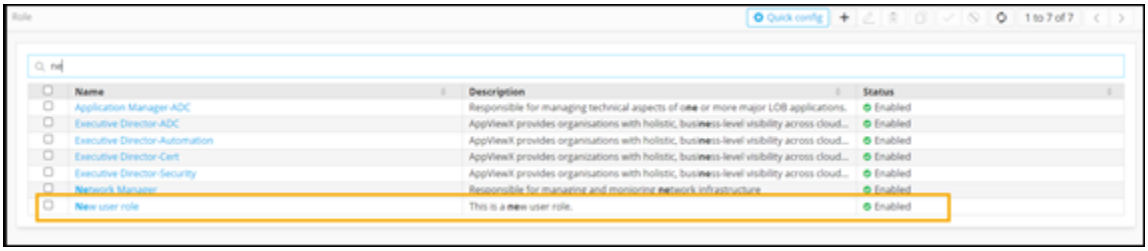
1. Hover over the form widget and click **(Edit)** icon.
2. In the **Form** pop-up window, under **Properties**, scroll down to **Form actions**.
3. Under Form actions, turn on the **Enable page redirect** toggle.
4. In the **Redirect URL** field, enter the URL of the page to be redirected to.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



5. Click **Save**.
6. To create a new role and redirect to the required page, click **Preview**.
7. Enter the **Role name** and **Description**.
8. Click **Submit**.

   You will be redirected to the **Role** page in the same tab.



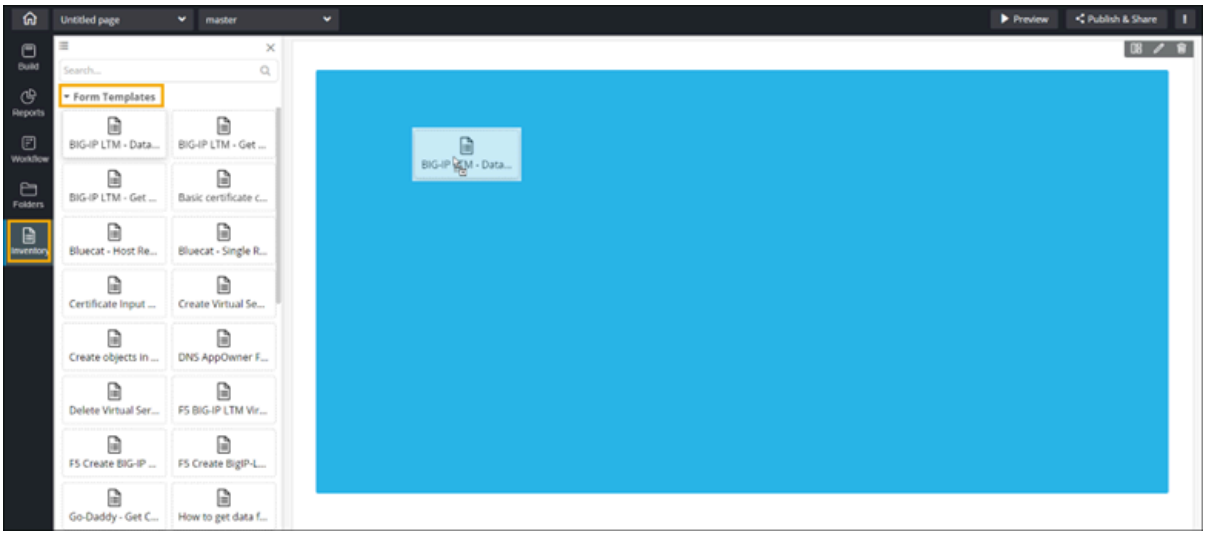
Using Form Templates

The **Inventory** section contains a set of prebuilt out of the box (OOB) Form Templates that you can add to your page for quick self-servicing.

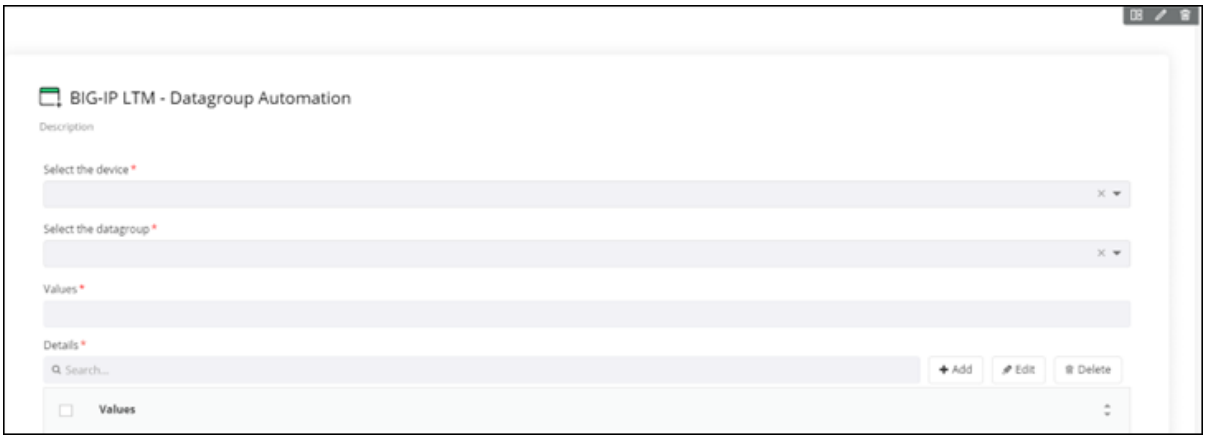
To use a Form Template:

1. Design a page.
2. From the **Inventory** section, expand **Form Templates**.
3. Drag and drop the required form template to the canvas.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The selected form template is added to the canvas. It can be used as is or modified as per your requirement.



Enabling Auto trigger

This feature allows you to define an auto trigger for a form field. Enabling auto trigger fetches data automatically to populate the form field(s). In case Auto trigger is disabled, a icon is displayed next to the form field. Clicking the icon will fetch the data from the database and populate the form field.

To enable auto trigger for a form field:

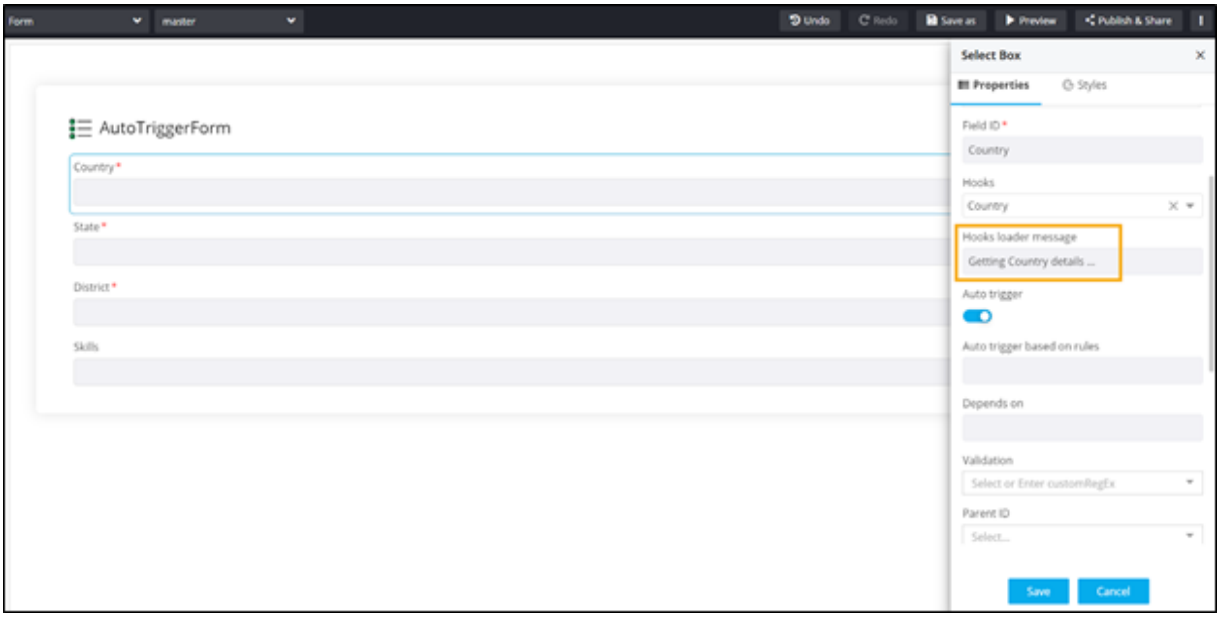
1. Design a form.
2. To edit a form field, hover your mouse over the field and click **(Edit)** icon.
3. In the form field window, under the **Properties** tab, define an appropriate hook for this field.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Once the hook is defined, the **Hooks loader message** field is displayed.

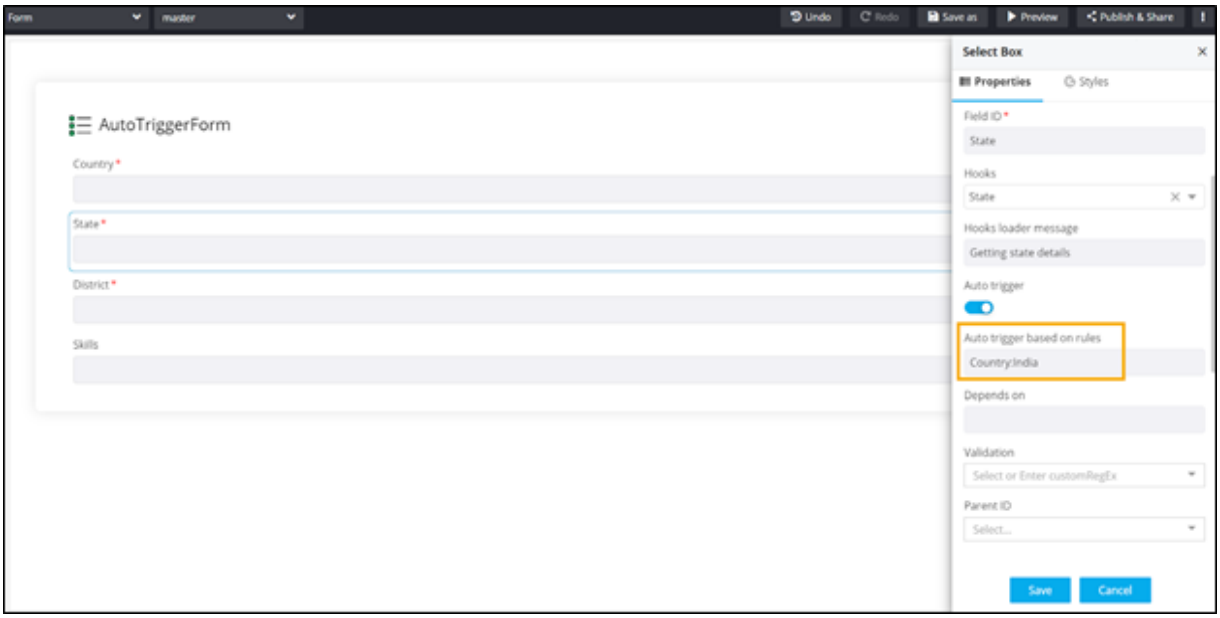
Note: For information on assigning hooks, click here.

4. Enter a message in the **Hooks loader message** field. This message is displayed while the form field is loading the field values.



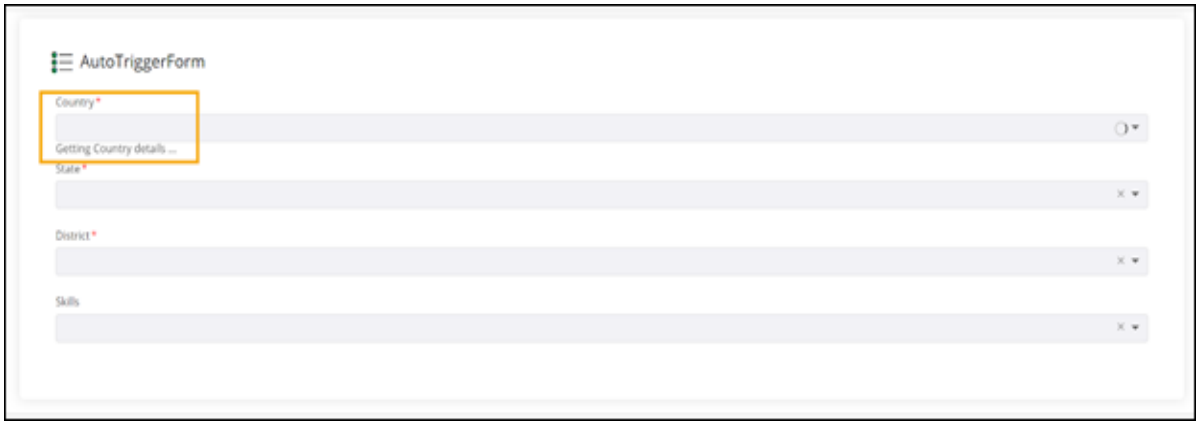
5. To enable **Auto trigger**, turn on the toggle.
6. In the **Auto trigger based on rules** field, define a rule for populating the next field in the form.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: This field is displayed only after a hook is assigned to the form field. Rules cannot be defined for the first field in the form.

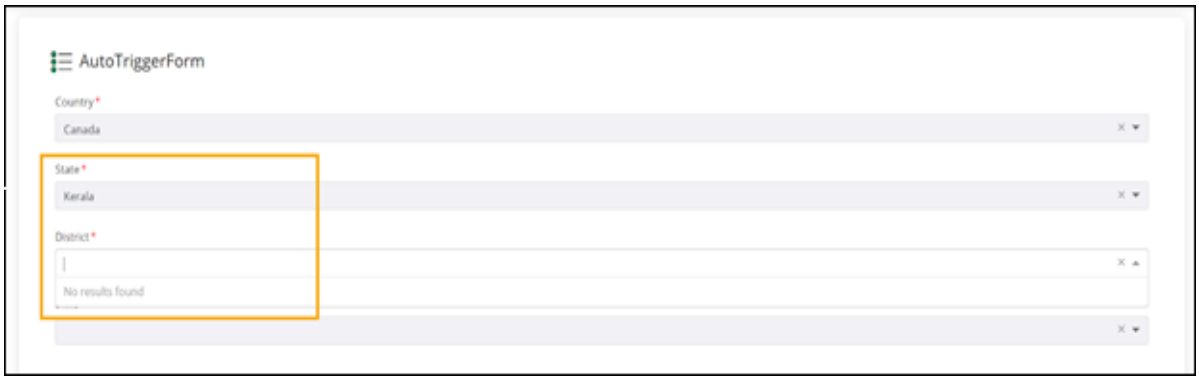
7. Define hooks for the other fields and enable **Auto trigger** for form fields, where required.
8. Add a Hooks loader message and define rules for required fields.
9. Click **Preview**.
   - A preview of the form opens in a new tab and the data for the first field is fetched automatically. The Hook loader message is displayed while the values are being fetched.



   - The values in the next form field are fetched automatically once a selection is made in the previous form field based on the rules defined.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

- Values in the next form field are not fetched/displayed if a selection other than the defined rule is made. For example, the rule “State:TamilNadu” has been defined for the District field. So, if any value other than TamilNadu is selected in the State field, then no results will be displayed in the District field.



- Once the value TamilNadu is selected in the State field, the values for the District field will be fetched automatically.

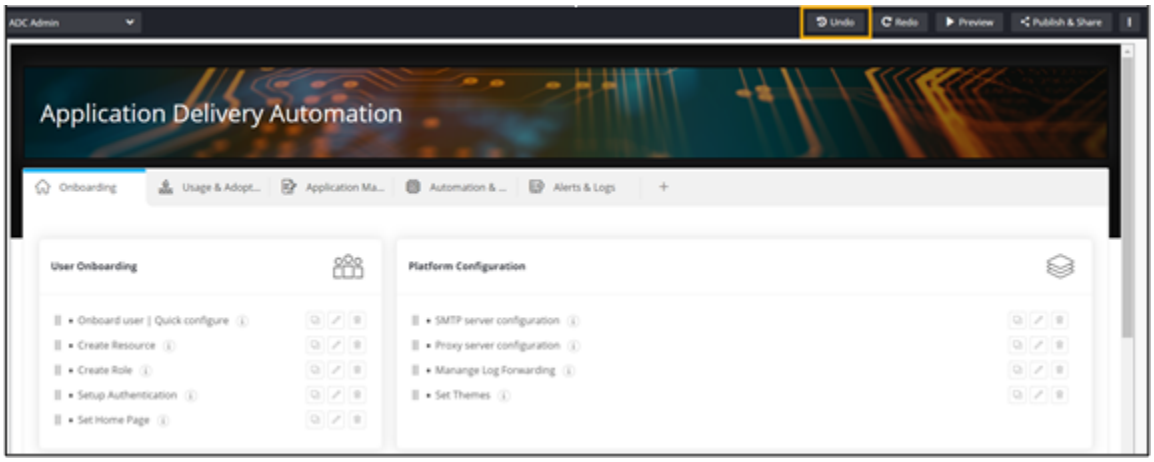
How to Undo/Redo actions

You can undo/redo any actions such as adding or deleting page elements.

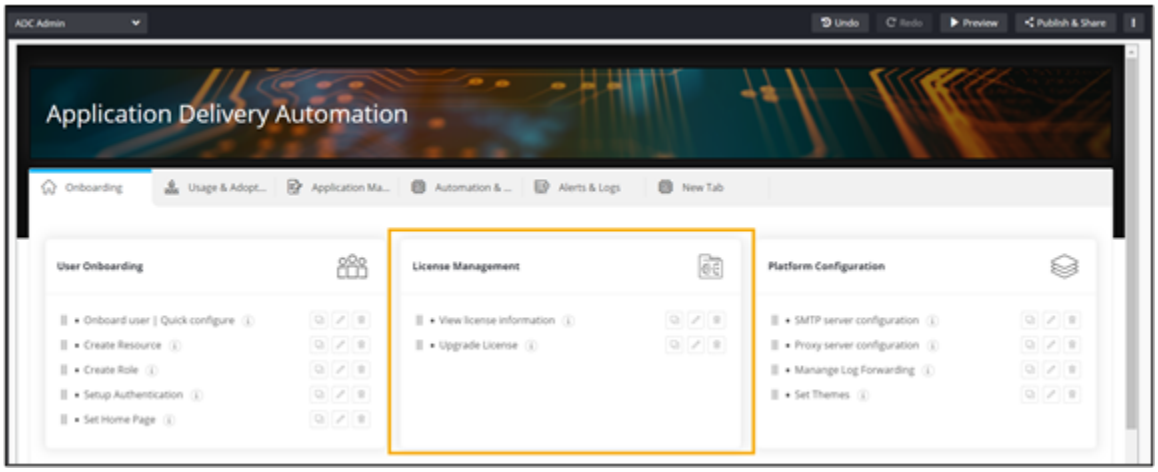
To undo an action:

1. Open a page in design mode.
2. From the menu on the top, click **Undo**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The deleted widget is displayed.



3. To redo the previous action, from the menu on the top, click **Redo**.
   The widget is deleted again.

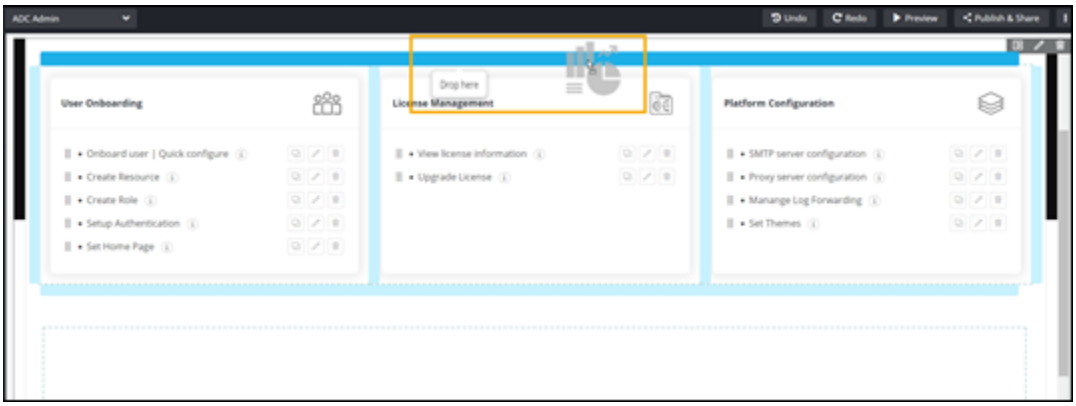
Moving widgets

You can modify the placement of the widgets on your page and move it around as per your preference.

To move a widget:

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

1. Hover your mouse over the widget and click **(Move)** icon.
2. Move the widget to any blue highlighted space in the build area.

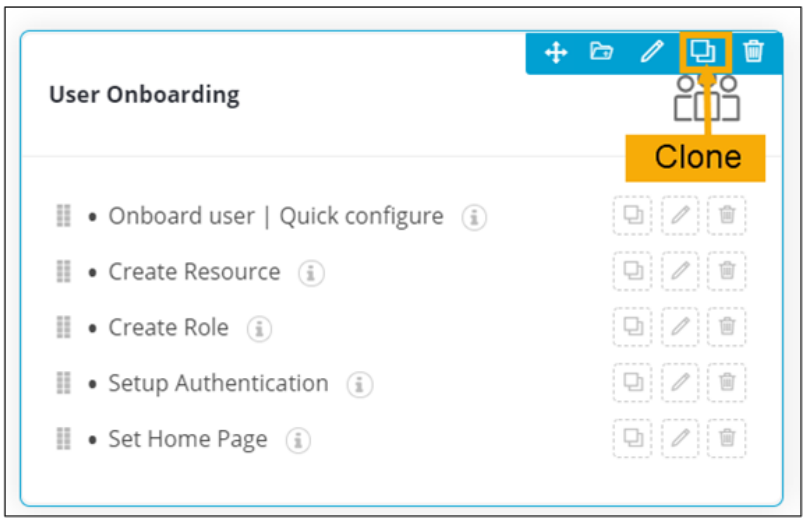


Cloning Widgets

You can clone a widget to have multiple widgets of the same component on your page, without having to drag and drop the components onto the page every time.

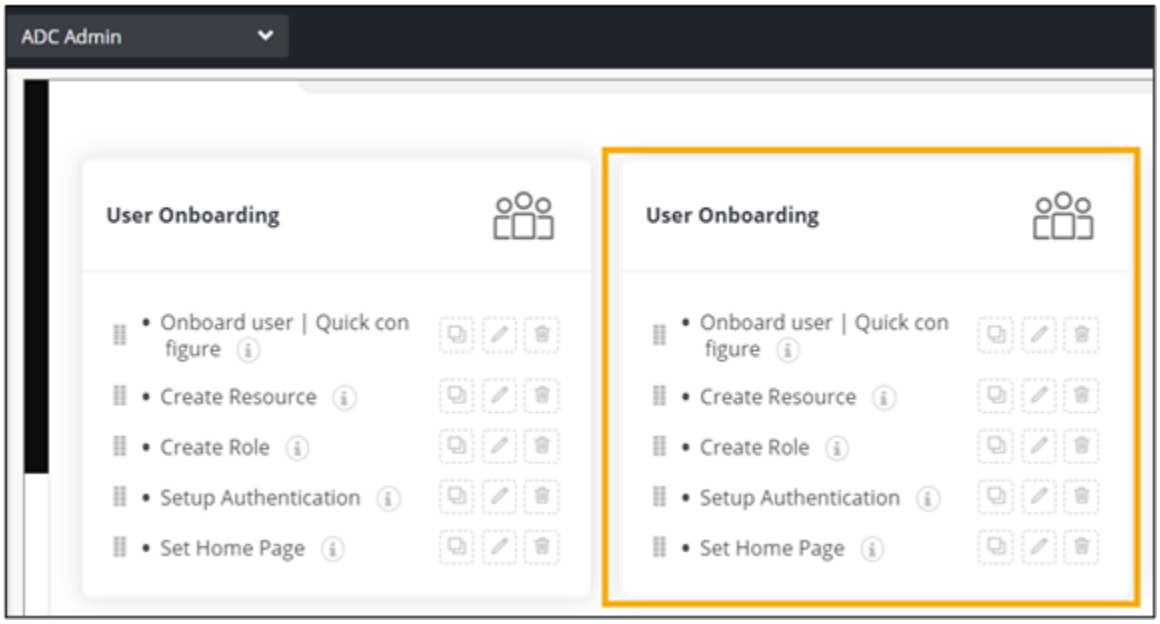
To clone a widget:

1. Hover your mouse over the widget.
2. Click **(Clone)** icon.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

A clone of the selected widget is displayed on the page. You can modify this cloned widget as per your requirement.



Version Control

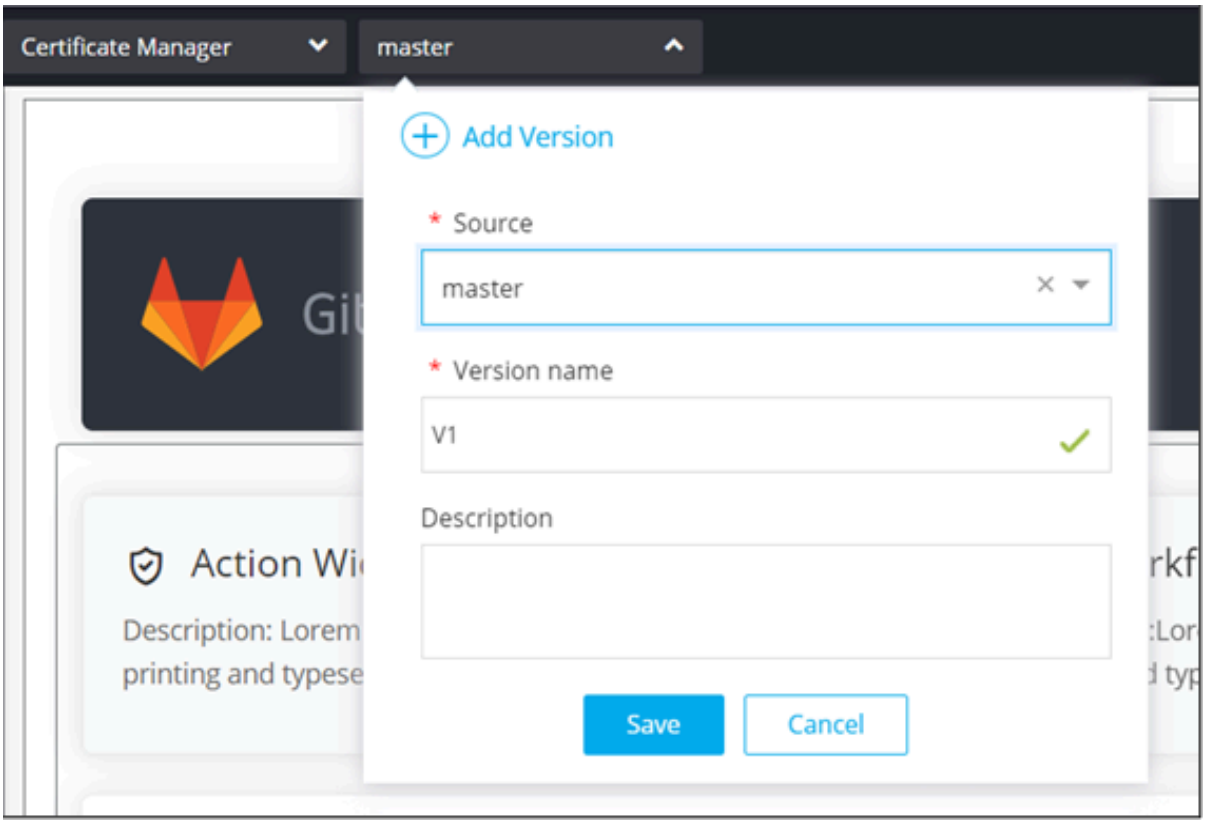
This feature allows you to design and have multiple versions of a catalog and enable one master version.

- Provision to version control the catalogs.
- Provision to edit/configure page elements in each version.
- Version control allows for tracking up to ten versions of the catalog.
- Provision to edit and delete versions.
- Provision to preview another version in a new tab/window while working on the current version.

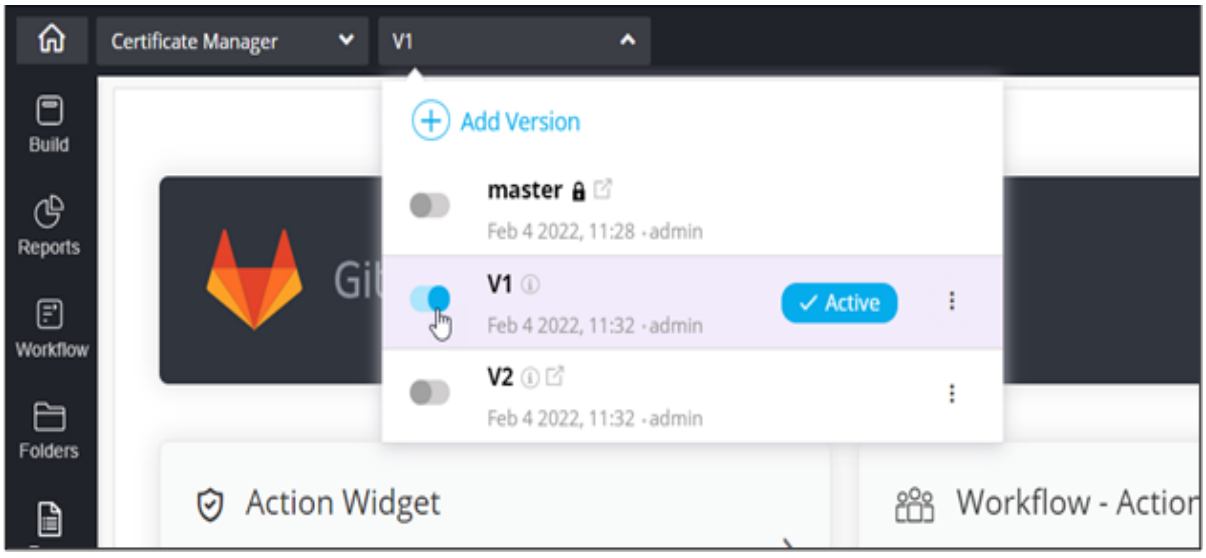
To add a new version:

1. From the **master** dropdown menu, select **Add Version**.
2. Select the **Source** and enter a **Version name**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

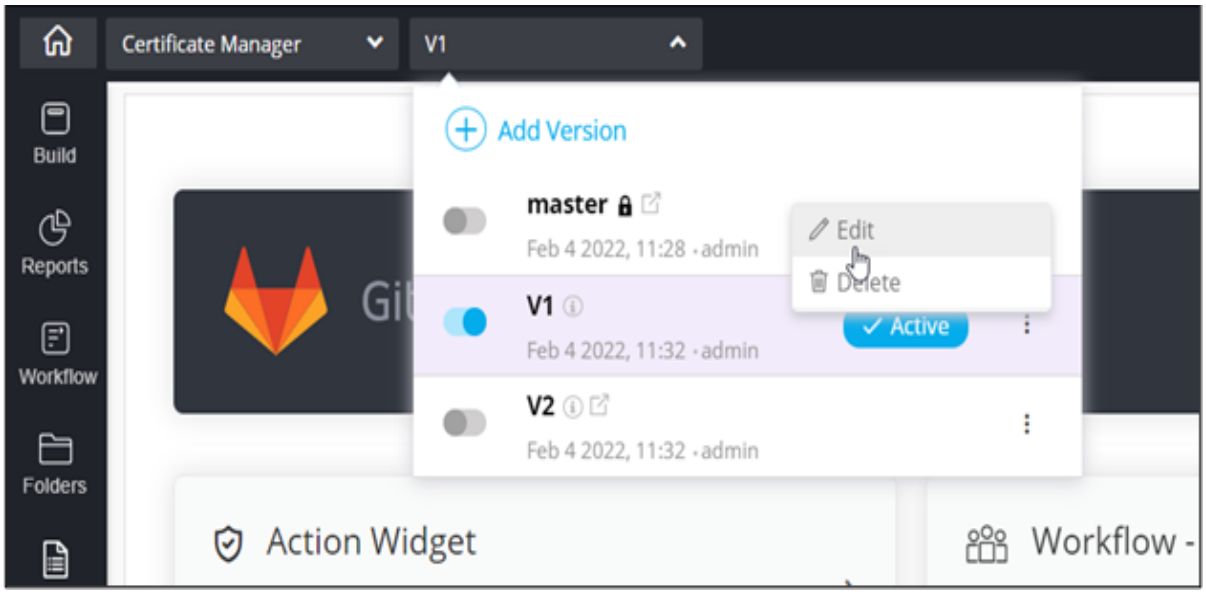


3. Click **Save**.
4. To mark a particular version as the active version for the catalog, enable the toggle next to it.

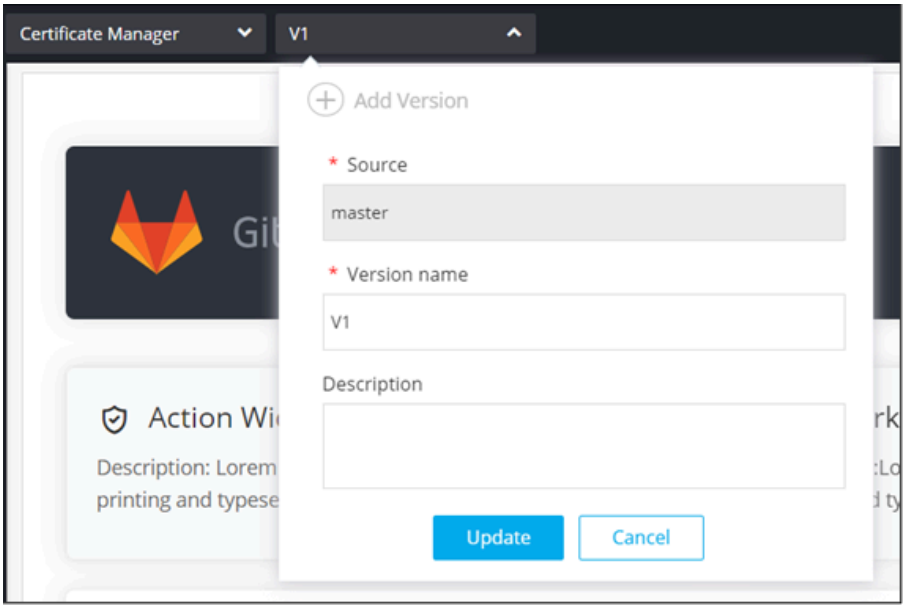


5. For more options, click ⋮.
6. To edit the version details, from the options displayed, click **Edit**.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



7. Edit the version details and click **Update**.



Important: The master version cannot be edited.

8. To delete the version, click ⋮.
9. From the options displayed, click **Delete**.

Important: The master version or any active version cannot be deleted.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Preview a Service Catalog

You can see a preview of your custom catalog before publishing it. Also, certain functionalities such as modifications in the workflow components are only available in the preview mode.

To preview a service catalog:

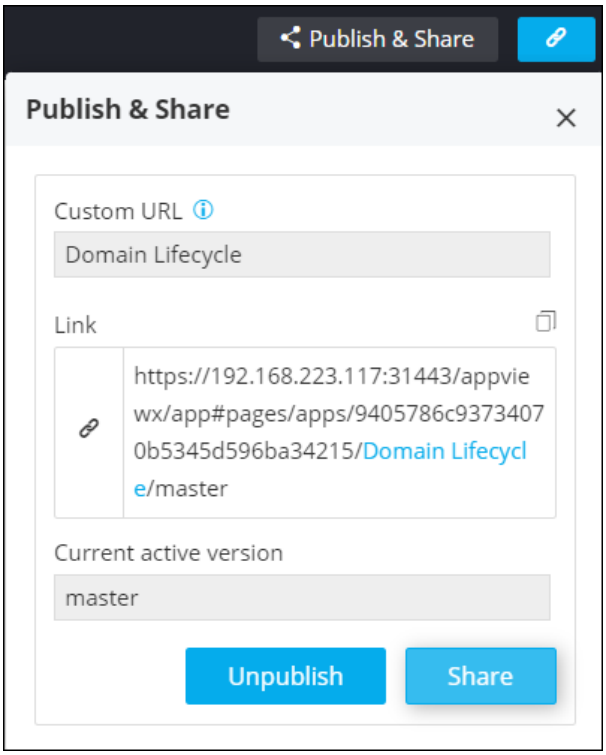
1. Open a catalog from the Design inventory.
2. From the top right corner of the screen, click **Preview**.
   A preview of the catalog opens in a new tab on your screen.

Publishing a Service Catalog

You can publish a service catalog once all the required components have been configured.

To publish a service catalog:

1. From the top right corner of the catalog, click **Publish & Share**.
   The **Publish & Share** pop-up window opens.



Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Field descriptions for Publish & Share

Field	Description
Custom URL	Allows you to enter a **Custom URL** for the catalog. **Note:** Page name is set as the default URL.
Link	Displays the link for the catalog. Click on the link to open your catalog in a new tab. To copy the link click **(Clone)** icon.
Current active version	Displays the current active version of the catalog. **Note:** This is a read-only field.

2. To publish your catalog, in the **Publish & Share** window, click **Publish**.
   The Published catalog is added to the **Published** Inventory.

Important: Only the current active version of the catalog will be published and added to the **Published** inventory.

Sharing a Service Catalog

You can share your published service catalogs with other users and/or user groups.

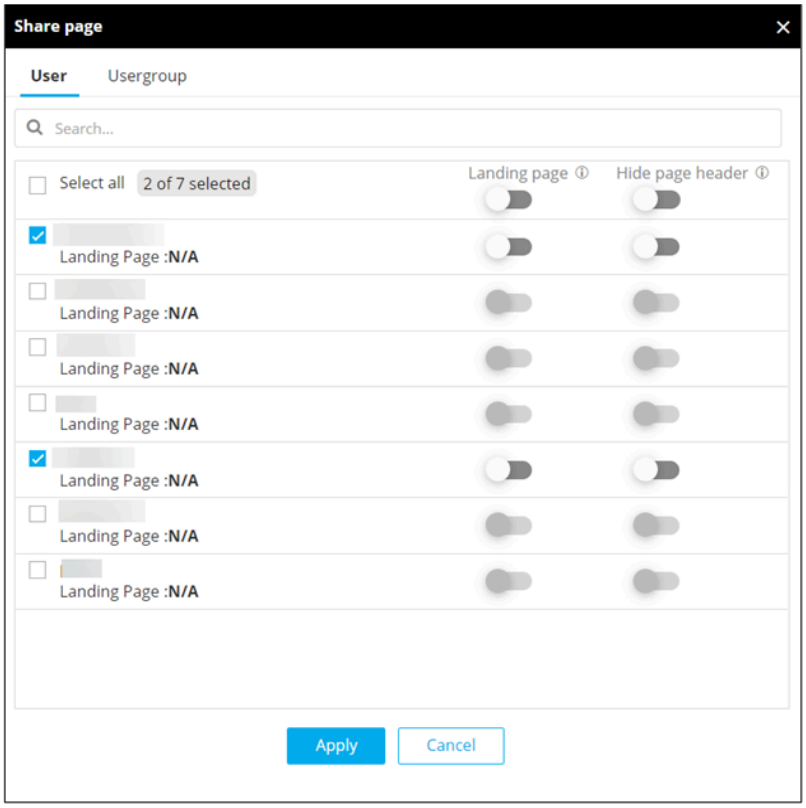
- Sharing a Service Catalog with a User(s)
- Sharing a Service Catalog with a User group(s)
- Scenarios

Sharing a Service Catalog with a User(s)

1. Open a catalog from the Design section.
2. From the top right corner of the screen, click **Publish & Share**.
3. In the **Publish & Share** window, click **Share**.
   The **Share page** pop-up window is displayed.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

4. To select the user with whom the catalog will be shared, in the **Share page** pop-up window, under **User**, select the checkbox next to the username.



Note: A catalog can be shared with multiple users.

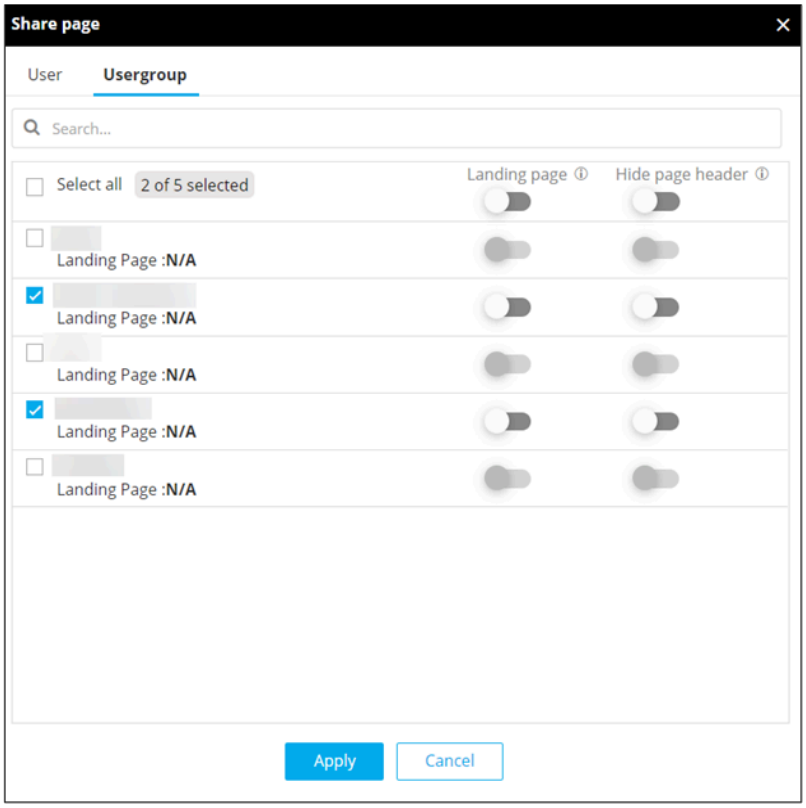
5. To share the catalog with selected user(s), click **Apply**.

Important: To share a specific version of the catalog, ensure it is the current active version.

Sharing a Service Catalog with a User group(s)

1. Open a catalog from the Design section.
2. From the top right corner of the screen, click **Publish & Share**.
3. In the **Publish & Share** window, click **Share**.
   The **Share page** pop-up window is displayed.
4. To select the usergroup with whom the catalog will be shared, in the **Share page** window, under **Usergroup**, select the checkbox next to the usergroup.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



Note: A catalog can be shared with multiple user groups.

5. To share the catalog with the selected user group(s), click **Apply**.

Important: To share a specific version of the catalog, ensure it is the current active version.

Scenarios

When a user creates a page and shares it with another user(s) in design mode,

- All actions such as renaming the page, adding/deleting components, changes in widgets and their properties and so on made by the user who created the page will be reflected for the user(s) with which the page has been shared.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Service Catalogs with Birthright Role

Birthright provisioning allows external users to log into AppViewX even when they have not been assigned to any user groups. By enabling birthright to a user group, when an external user logs into AppViewX, they will see the landing page of the user groups for which Birthright has been enabled.

If an external user has been assigned to a user group then,

- If that user group does not have a landing page set up, then the landing page associated with the group for which Birthright is enabled will be the landing page for this user.
- If that user group has a landing page set up, then the landing page that appears when the user logs in will be dynamic. However, any further changes to the landing page settings will impact the landing page of this user. The latest page set as the landing page becomes the landing page of this user when they login.

• Limitation

Limitation

If the birthright provisioning is changed from one user group to another, for example, if the birthright was enabled for the ‘Admin’ user group, but now it has been enabled for the ‘end user’ user group, there is no change to the landing page for the user, i.e. the landing page of the ‘end user’ user group will not be updated.

Cloning a Service Catalog

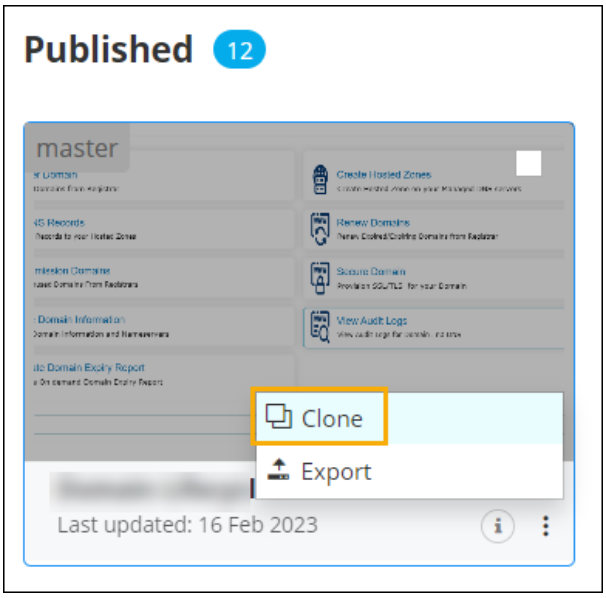
You can clone catalogs that are in the Published inventory as well as in the Design section.

• Cloning a Published Catalog
• Cloning a Catalog in Design mode

Cloning a Published Catalog

1. On the **Published** page, click on the page to be cloned.
2. From the options displayed, click **Clone**.

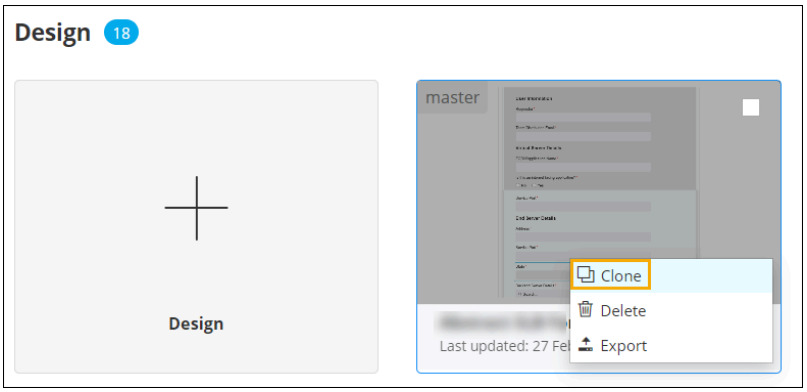
Copyright © 2025 AppViewX, Inc. All Rights Reserved.



The cloned catalog opens in the same tab and a copy of the cloned catalog is added to the **Design Inventory**.

Cloning a Catalog in Design mode

1. On the **Design** page, click ⋮ on the page to be cloned.
2. From the options displayed, click **Clone**.



The selected catalog opens in the same tab and a copy of the cloned catalog is added to the **Design** inventory.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Setting up the Landing Page

The Landing page is the page that appears first on the screen when a user logs into AppViewX. Only a Published service catalog can be shared and set as a landing page. This Landing page can be also accessed by clicking the AppViewX logo on the top left corner of the computer screen.

Important: When setting the Landing page, only the current active version of the catalog will be set as the Landing page.

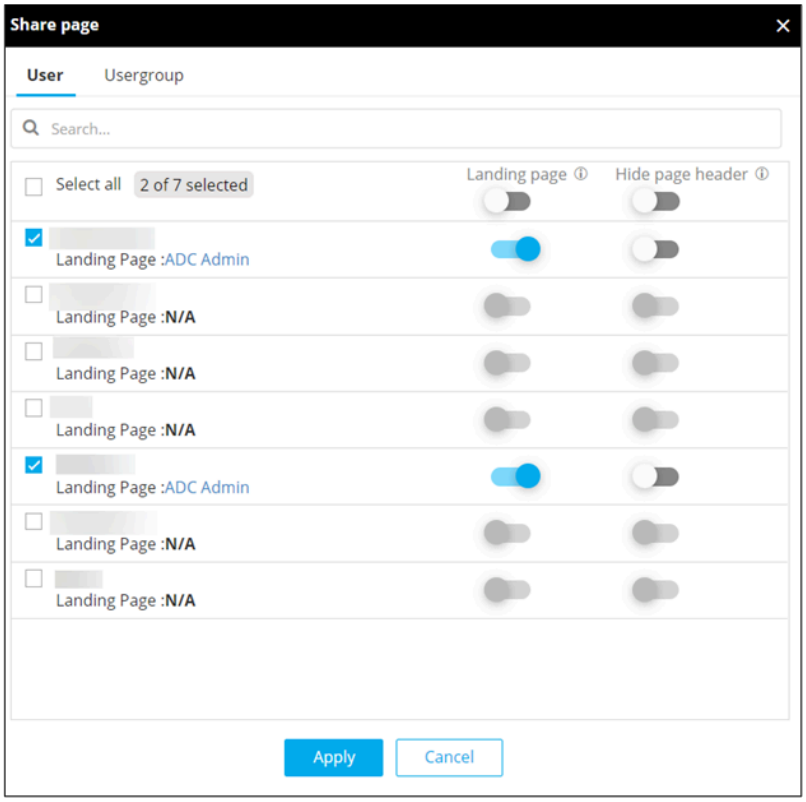
- Setting up the Landing Page when Publishing/Sharing
- Setting up the Landing Page under User Preferences

Setting up the Landing Page when Publishing/Sharing

To set the current page as a landing page for certain user(s) and/or user group(s):

1. Open a Published page.
2. From the top right corner of the screen, click **Publish & Share**.
3. In the **Publish & Share** window, click **Share**.
4. In the **Share page** window, select the user(s) for which the current catalog is to be set as the Landing page.
5. To set the current catalog as the landing page for selected user(s), in the **Share page** window, under **Landing page,** turn on the toggle for the selected user(s).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. To implement the selected settings, click **Apply**.

> **Note:** The current active version of the catalog will be set as the Landing page for the selected users.

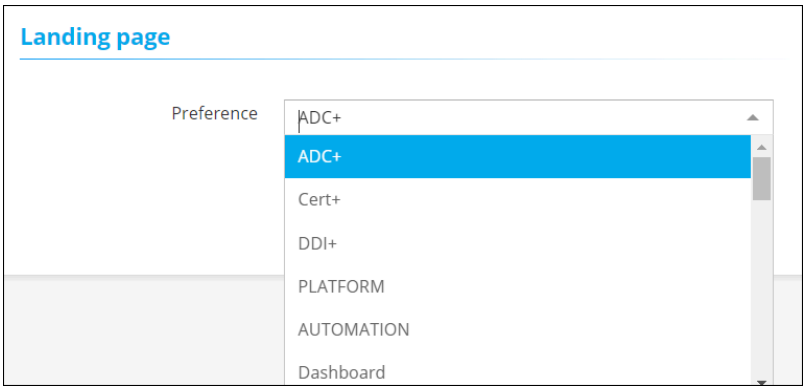
Setting up the Landing Page under User Preferences

You can also set a specific page as your landing page by modifying user preferences.

To set landing page under user preferences:

1. From the top right corner of the screen, click **(Profile)** icon.
2. From the options displayed, select **User preferences**.
3. On the **User preferences** page, from the drop down in the **Landing page** section, select the catalog that you want to set as the landing page.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



4. To set the selected catalog as your landing page, click **Save**.

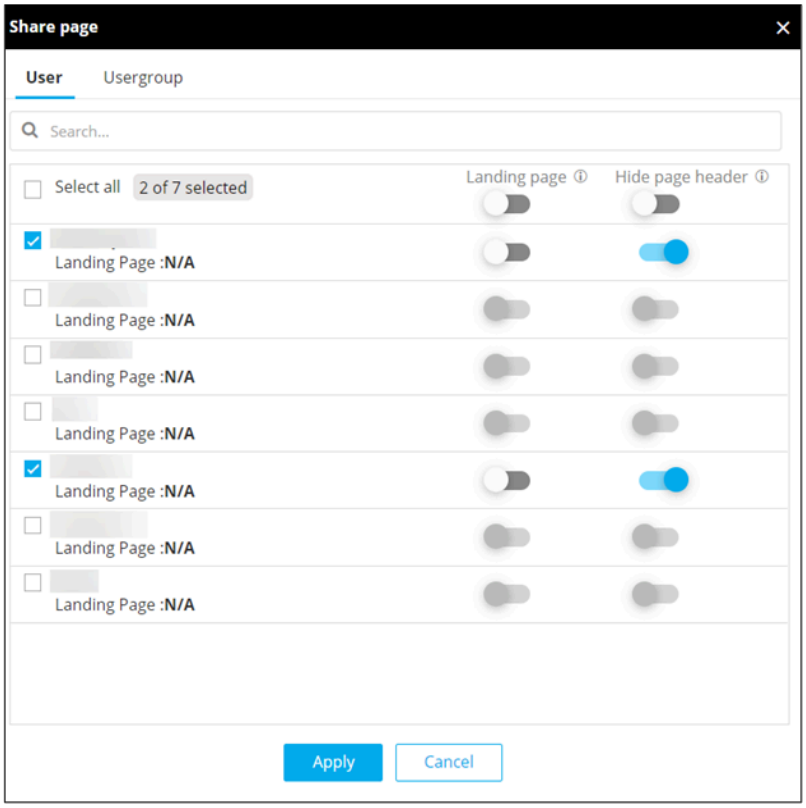
Hiding the Page Header

You have the option of hiding the AppViewX header on your published catalogs. This can be done while sharing a published catalog with user(s) and/or usergroup(s).

To hide the page header:

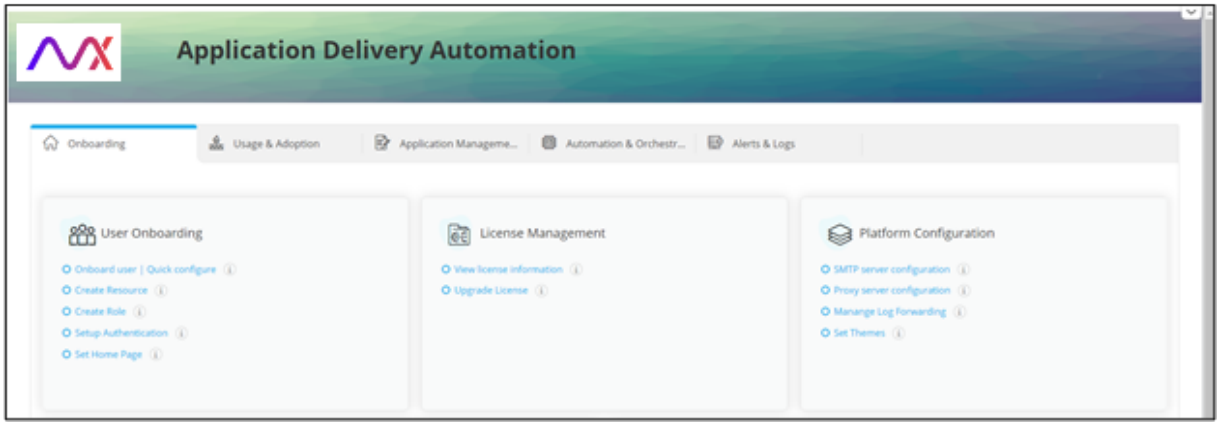
1. Open a Published page.
2. From the top right corner of the screen, click **Publish & Share**.
3. In the **Publish & Share** window, click **Share**.
4. In the **Share page** window, select the user(s) from the list.
5. To hide the page header for the selected user(s), under **Hide page header**, turn on the toggle for the selected user(s).

Copyright © 2025 AppViewX, Inc. All Rights Reserved.



6. To implement the settings, click **Apply**.

   The AppViewX header is hidden.



Note: The Hide page header option is independent of whether the catalog is set as the Landing page for a user or not. The page header will be hidden even when the Published catalog is accessed with its URL.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

7. To show the application header, from the top right corner of the screen click .

8. To collapse the application header, from the top right of the screen, click .

Troubleshooting

Following are the common issues and their solutions:

Issue	Cause	Solution
Service Catalog module does not load	• avx-web-visual-page-builder pod is down. • avx-subsystems-automation pod is down.	Restart the pods and refresh the catalog.
Design canvas throws error	• avx-web-visual-page-builder pod is down. • visual_page_builder_toolbox is not available as a web plugin/addOn.	• Restart avx-web-visual-page-builder pod. • Deploy the visual_page_builder_toolbox in the required location.
Catalog actions (Create, Publish, Share etc.) not enabled or available	The user might not have permission for these Service catalog actions enabled.	Enable the required Service catalog action permissions for the user's role.
Landing page not set appropriately for shared users	• The catalog might have been shared before it was set as the landing page. • Landing page setting might be overridden in page share tab for user/user group or application user preference.	• The catalog has to be shared after setting the landing page. • Cross verify landing page setting in share tab and application user preference.

Copyright © 2025 AppViewX, Inc. All Rights Reserved.

Following are the common issues and their solutions: (continued)

Issue	Cause	Solution
Improper data or error in widgets	Discrepancies in data sent/received from server.	Analyze/Share the Service Catalog related logs listed here with dev team: • webApp.log • visual_page_builder_engine.log • avx-subsystems.log

Copyright © 2025 AppViewX, Inc. All Rights Reserved.