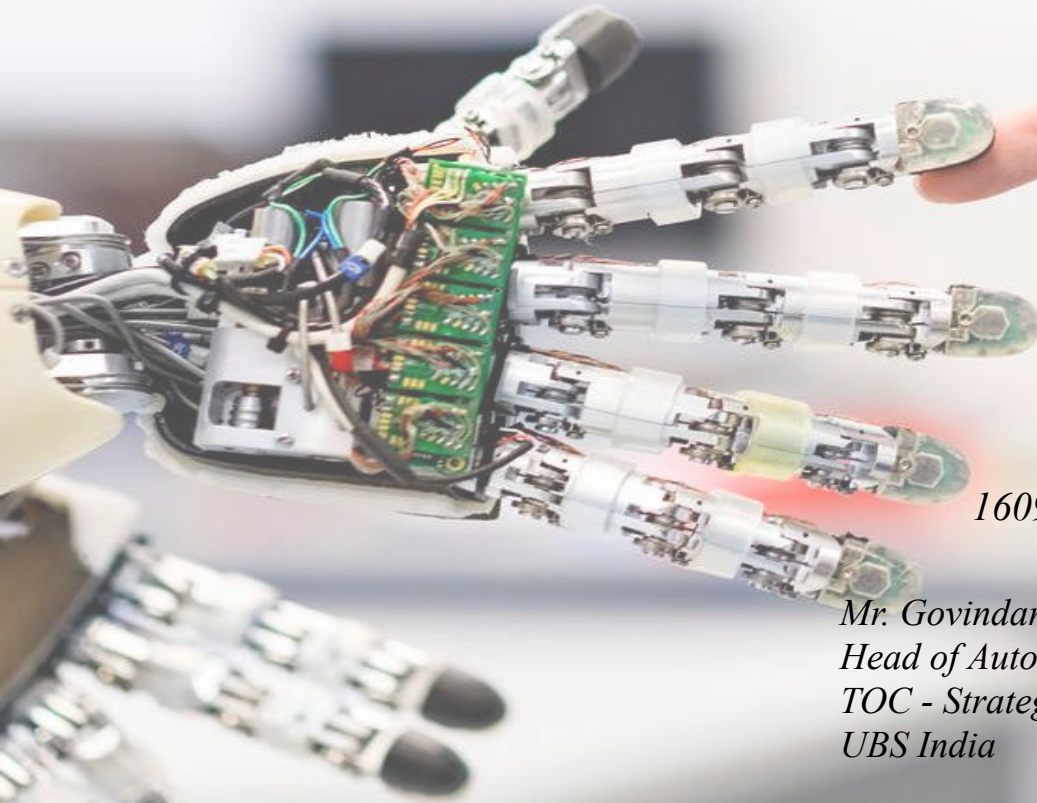




# *Ticket Enrichment using Machine Learning & Automation Catalog Development*



*Prepared by,  
B N S Vishal  
160911122, IT-'A', MIT, Manipal  
Under the Guidance of*

*Mr. Govindarajan VS  
Head of Automation CoE  
TOC - Strategy and Transformation  
UBS India*

*Mrs. Anuradha Rao  
Asst. Professor Senior Scale  
Dept. of ICT  
M.I.T. Manipal*

## ***Introduction***

**Automation** is the creation of software and systems to replace repeatable processes and reduce manual intervention. Regular health checks of remote servers, resolution of errors in applications and other all repeatable and mundane tasks could be automated.

The following two projects aim at delivering standardized and time & cost saving solutions to the automation developers and support analysts across the organization -

1. *Ticket Enrichment using Machine Learning*
2. *Automation Catalog*

# *The Automation Catalog*

The Automation Catalog is a highly visible website across the organization.

It is conceived as a data store to record all live automations in TOC deployed through strategic platforms - IPCenter and Automation Anywhere and non-strategic platforms - Python, Shell, Perl etc. in order to promote 'reusability' of standardized and commoditized (off the shelf readily deployable) automations to save development time and to reduce risks and costs that may incur.

## **Tools and Development Environment-**

The website is built using Angular 8.

IDE - IntelliJ Idea Ultimate 2019

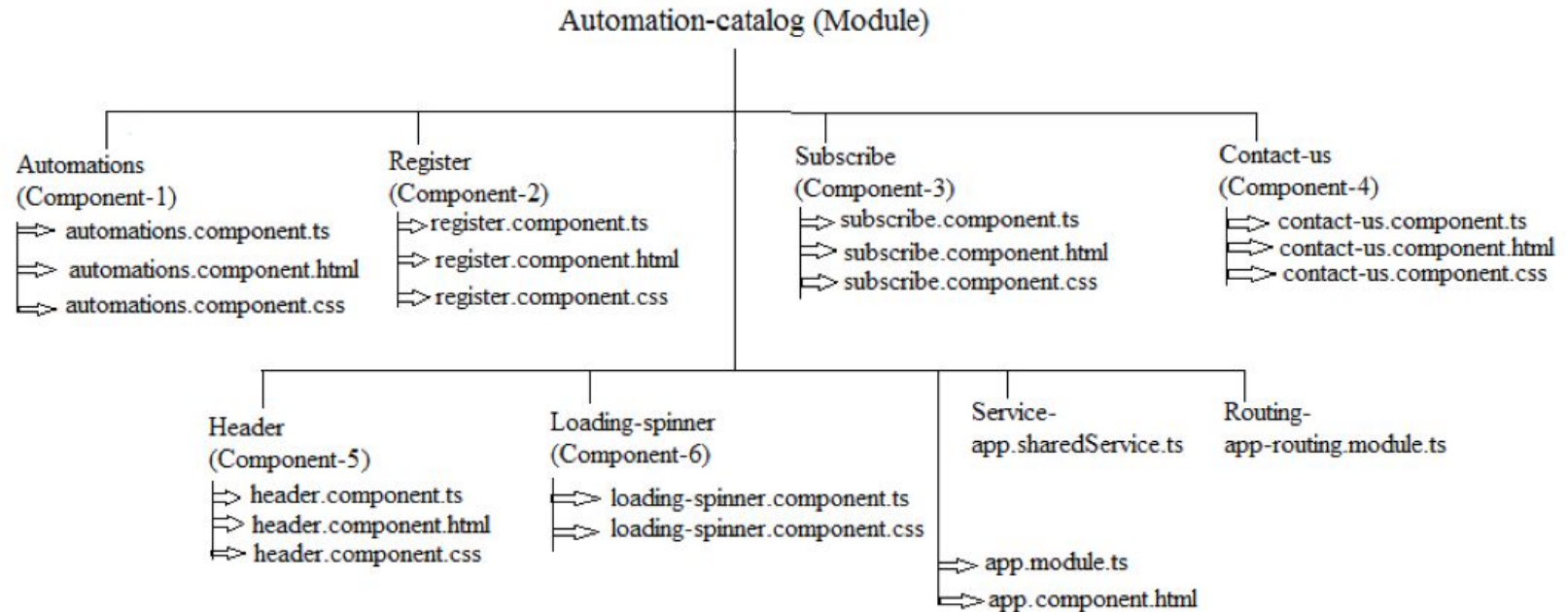
Node Version - 10.16.9

NPM Version - 6.9.0

Angular CLI Version - 9.1.4

Bootstrap Version - 3.3.7

## *Application Structure/ Methodology*







## Browse Catalog

### By Source

- ServiceNow
- Netcool

### By Type

- Incident
- Run
- Query
- Request
- Governance and Op Design
- Package or Deploy
- Production Assurance
- Management Support
- Risk and Controls
- Continuous Improvement

### By Tools

- IPCenter
- Automation Anywhere

### What is the Catalog?

The catalog enables the acceleration of your digital transformation, by providing access to a library of code for automation solutions that you can re-use for automation requirements in your sector. The content of the catalog is continually expanding.

### How does it Help Me?

Re-using the existing code saves time in the development process, and helps standardize across the sectors. Challenges in design and approach have already been worked out, to provide a successfully operating solution.

### How do I use it? In three easy steps!

**Browse** the catalog for an existing automation to re-use – look for one that is the same or similar to your requirement

**Build** your automation using the catalog code, taking it through the normal development lifecycle

**Release** your automation to production – the catalog will be automatically updated once you have implemented

### Can I share my own Automations?

Yes, click on 'Register Your Automation' in menu and submit the details. It will be added to the catalog after approval.

## What's New?

### Execution Stats

Check out the execution stats of the automations listed from most number of executions to least.

14211



Visitors

### Link to source code

Links to the source code of non-strategic automation components is now available in their respective tabs.

551



Automations Onboarded

416



IPC Automata

117



AA Components

18



Script Automations

### Subscription

Subscribe to the automation catalog and get notified when new automations get added.

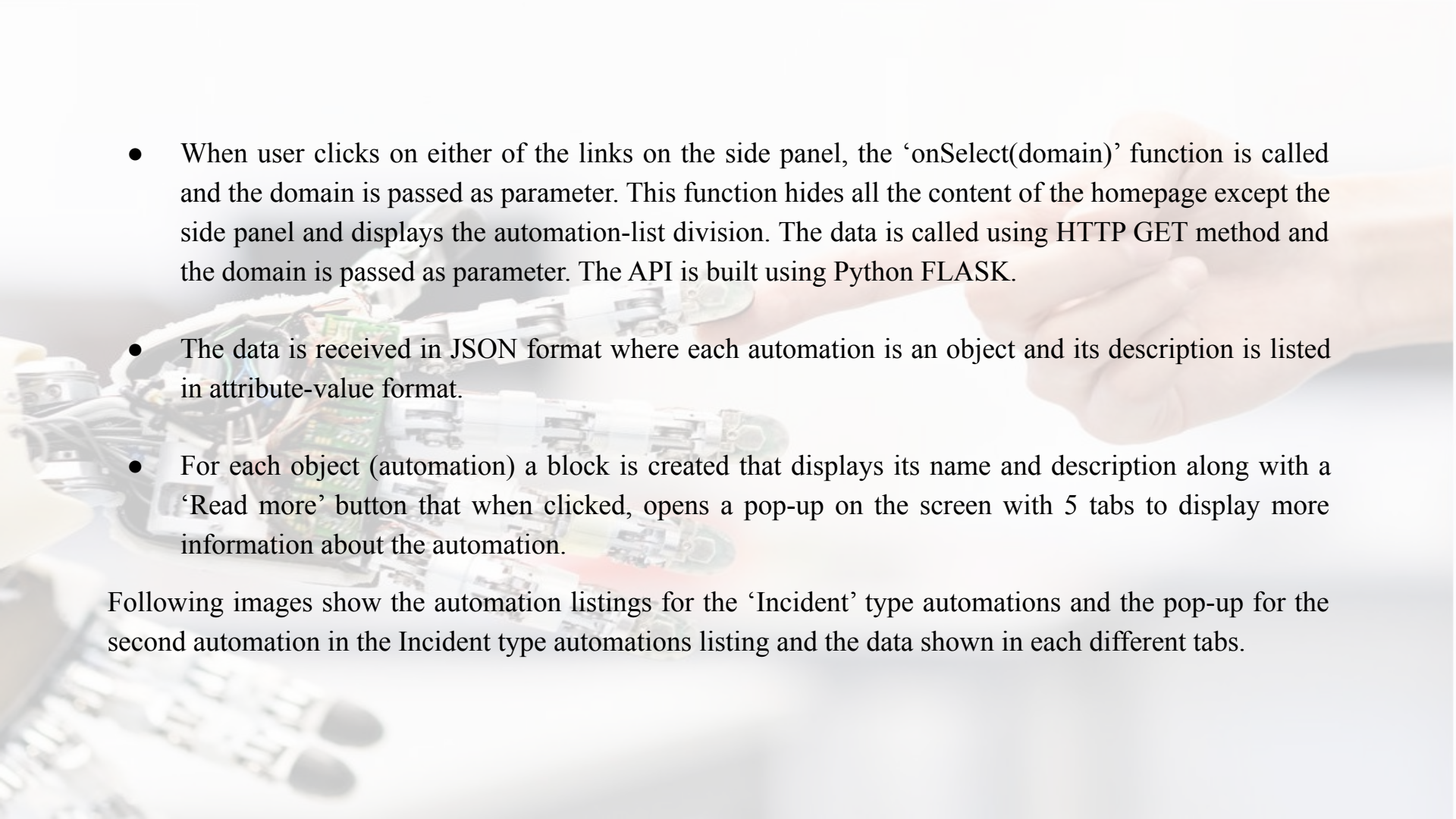
## *Automations Component*

This component displays the homepage as well as the automation listings. Initially, the homepage is loaded whenever users visit the website.

- The homepage is the main page that describes the ‘what, why and how’ of the catalog. It also displays some statistics about the site in the form of widgets on the bottom of the page.



- Visitors count and the count of total automations onboard are done in the backend and the data is called using the HTTP GET method. The APIs are built using Python FLASK.

- 
- When user clicks on either of the links on the side panel, the 'onSelect(domain)' function is called and the domain is passed as parameter. This function hides all the content of the homepage except the side panel and displays the automation-list division. The data is called using HTTP GET method and the domain is passed as parameter. The API is built using Python FLASK.
  - The data is received in JSON format where each automation is an object and its description is listed in attribute-value format.
  - For each object (automation) a block is created that displays its name and description along with a 'Read more' button that when clicked, opens a pop-up on the screen with 5 tabs to display more information about the automation.

Following images show the automation listings for the 'Incident' type automations and the pop-up for the second automation in the Incident type automations listing and the data shown in each different tabs.



## Browse Catalog

### By Source

- ServiceNow
- Netcool

### By Type

- Incident
- Run
- Query
- Request
- Governance and Op Design
- Package or Deploy
- Production Assurance
- Management Support
- Risk and Controls
- Continuous Improvement

### By Tools

- IPCenter
- Automation Anywhere

## Incident

### IIS\_Application\_Alerts-v1.15



Enhancement : To set escalation level to L1 for alerts originating from servers operated for GLAM AA24951 AT7730 MIRO and Matcher Updated for JIRA - AUTOIS-

[Read more](#)

### Dynamic Correlation



IPCenter automatically correlates similar patterns raised in netcool within a specific timeframe, raising one SNOW ticket only and adding any future occurrences within the

[Read more](#)

### serviceCrashAlerts-v1.3.02



AA24951 AT7730

[Read more](#)

### High Memory Utilization



Resolution of these tickets involves checking & analyzing memory utilization and reporting to the server owner in case of frequent high memory utilization of any

[Read more](#)

### Memory Utilization



Enhancement of AUTOIS-10 High memory utilization

[Read more](#)

### DbaaS Alerts - Memory Utilization



Takes in DBaaS memory alerts and suppress them.  
Agent/Domain=DBaaS - GBL,  
AlertGroup/ParameterName=DBaaS Infra: Host,

[Read more](#)




## Automation Description Tab-

**Dynamic Correlation** Close

Automation Description	Automation Details	Reusability considerations	Using existing data sources	Execution Stats
<p><b>Category:</b> Incident</p> <p><b>Access Automation via:</b> <a href="http://a301-6737-8537.sng.swissbank.com:5001/1">http://a301-6737-8537.sng.swissbank.com:5001/1</a></p> <p><b>Description:</b> IPCenter automatically correlates similar patterns raised in netcool within a specific timeframe, raising one SNOW ticket only and adding any future occurrences within the set period into the work log of the master ticket. The incident short description is update to show that Multiple tickets are logged within this master ticket</p>				

## Automation Details Tab-

**Dynamic Correlation** Close

Automation Description	Automation Details	Reusability considerations	Using existing data sources	Execution Stats
<p><b>Scope of the automation :</b></p> <p>Application Alerts</p> <p><b>Date of automation go -live :</b></p> <p>None</p> <p><b>Automation Lead who has overseen the automation :</b></p> <p>Michael Arrigan</p> <p><b>SMEs associated with the automation :</b></p> <p>Vikram Maturi / Praveen Venkataram</p> <p><b>Sectors using the automation :</b></p> <p>CC, IB, WMA</p> <p><b>Platform on which automation is done :</b></p> <p>IPCenter</p> <p><b>Toolset on which automation is done :</b></p> <p>IPCenter</p> <p></p>				

## Reusability Considerations Tab-

**Dynamic Correlation** Close

Automation Description

Automation Details

Reusability considerations

Using existing data sources

Execution Stats

**Assessment of reusability :**

High

**Portability considerations:**

Dynamic correlation only correlates application alerts, not autosys job failures

**Technology Limitations:**

If too many alerts are thrown within a short period of time <2 mins dynamic correlation cannot detect this as a duplicate as the SNOW DB IPC queries is not updated yet

**No. of times reused:**

TBD

**Known bottlenecks:**

NA

## Using existing data sources Tab-

**Dynamic Correlation** Close

Automation Description

Automation Details

Reusability considerations

Using existing data sources

Execution Stats

**Domain Automata:**

INC:BAS:FRCORC:ApplicationAlertsGeneric-V1.4

**Model JIRA reference:**

AUTOIS-18136

**Other associated JIRAs:**

NA

**Links and attachments:**

NA

## Execution Stats Tab-

**Dynamic Correlation**Close

Automation Description

Automation Details

Reusability considerations

Using existing data sources

Execution Stats

**Total Executions:** 101421

**Successful Executions:** 100385

**Failure Executions:** 18

**Assisted Executions:** 1018

# Subscription Page



Technology Operations Center - *Automation Catalog*

[Home](#)[Register Your Automation](#)[User Guide<sup>pdf</sup>](#)[Subscribe](#)[Contact Us](#)

## Subscribe to the Catalog

*Subscribe to get informed everytime new automations are added to the catalog!*

Please provide your email-id\*

[Subscribe](#)[Clear](#)[Close](#)



## *Subscription Component - Pseudo Code*

```
if(email does not end with '@ubs.com') {  
    An alert message pops up requesting user to enter only UBS email.  
}  
else {  
    1. Retrieve all subscribed emails from the database server using HTTP GET method and  
       store the list in an array.  
    2. Do a linear search to check if the email entered by the user is present in the list or not.  
  
    if(email is present in the array) {  
        An alert message pops up with the message - 'You have already subscribed to the  
        Catalog.'  
    } else {  
        1. Email entered is enclosed in the body of the request message of the HTTP POST  
           Method  
        2. Alert message pops up displaying the message - 'Thank you for subscribing to the  
           Catalog!'.  
    }  
}
```

# Automation Registration Page



## Register your Automation to the catalog

*(Your Automation will be added to the catalog after approval)*

Email-id\*

Enter your email-id...

Automation Name\*

Enter the name of the automation...

Reference id\*

Enter the JIRA/Story ID...

Category

Please select category

Reuseability Assessment

Select assessment range



Description\*

Brief about your automation in no more than 4000 words..

Scope of Automation\*

Health Checks...

Automation Platform\*



**Portability Consideration\***

Prerequisite before using your automation..

**Technology Limitations**

Technology limitations for adopting the automation

**Links or Attachments**

Knowledge article or document link..

**Automation Lead Name\***

**SME/Developer Name\***

**Automation-Go-Live Date\***

mm/dd/yyyy

**Automation Sector\***

**Tags\***

Please provide up to 3 tags separated by comma

© 2020 Technology Operations Center, All Rights Reserved

- All required fields marked with asterisks must be filled for the 'Submit' button to be enabled.
- The data is bound as a JSON object and passed as a parameter in the HTTP POST method and an alert message pops up that says - "Thank you for registering your Automation to the Catalog. The automation will be added after review by the central team."
- The details are then emailed to the central team for review.

# Contact Us Page



## Contact us for support or to give feedback

From\*

Message\*

Send


Clear


Close

- The 'From email' here too must be of '@ubs.com' domain.
- Data ('From email' and 'Message') is similarly bound as JSON object and passed as parameter in the HTTP POST method.
- Central team receive the data and can resolve the queries by contacting the user.



# Search Function

**Technology Operations Center - Automation Catalog**

health check

HomeRegister Your AutomationUser Guide<sup>24/7</sup>SubscribeContact Us

### Browse Catalog

#### By Source

- ServiceNow
- Netcool

#### By Type


- Incident
- Run
- Query
- Request
- Governance and Op Design
- Package or Deploy
- Production Assurance
- Management Support
- Risk and Controls
- Continuous Improvement

#### By Tools

- IPCenter
- Automation Anywhere

### Search results displayed for keyword: health check

#### Automated Feed Arrival Checks




Automation to handle feed delays related batch failures. IPCenter will handle the batch failure alerts by checking the required feed files. Incenter will raise a ticket with

[Read more](#)

★★★★★

#### Automated Weekend HealthChecks




Automation Anywhere bot to do application health checks (UI) and ensure functionalities of the applications work as expected

[Read more](#)

★★★★★

#### Automated batch maxrun monitoring and ticket closure




IPC Automation verifies the job status and determines the average runtime by checking the last 3 runs (+additional

[Read more](#)

★★★★★

#### Home Directory Deletion Check and Ticket Closure




The robot checks if the home drive has been deleted. If yes, then closes the respective ticket.

[Read more](#)

★★★★★

#### OBS Sunday Checks




OBS Sunday Checks

[Read more](#)

★★★★★

#### Application Heath Check UI automation



This application health check is to verify critical application s of HR Stream to ensure smooth

[Read more](#)

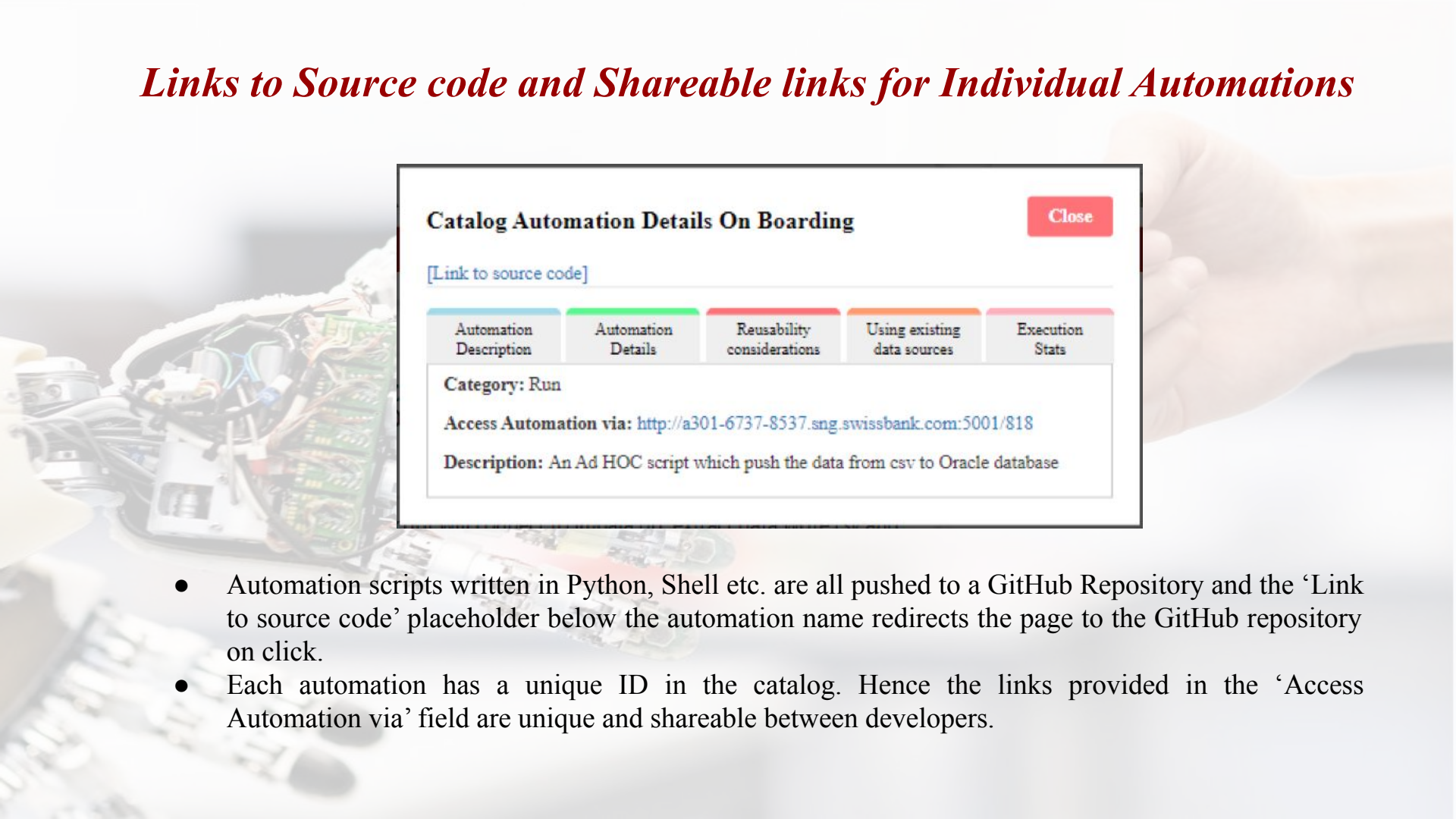
★★★★★

© 2020 Technology Operations Center, All Rights Reserved

## *Angular Service*

- The header component has an input field for searching automations.
- Because the header and the main page are two different components, Angular Service is used for passing the search query from the header to the automations component.
- Angular Subject is defined in the Service. It is a special type of Observable that allows data to be multicasted to many observers.
- The form in the Header component is the observee and the Automations component is the observer.
- Whenever a user enters a query, a 'sendMessage(query)' function is called and the subject is fed with a new value.
- The Automation Component that is subscribed to observer of the subject - 'getMessage()' function' written in the Service, 'observers' that the subject has received a new value, calls a 'doSearch(query)' function that retrieves the data from the server using the HTTP GET method and displays the results.

## *Links to Source code and Shareable links for Individual Automations*



**Catalog Automation Details On Boarding** Close

[\[Link to source code\]](#)

Automation Description	Automation Details	Reusability considerations	Using existing data sources	Execution Stats
<b>Category:</b> Run				
<b>Access Automation via:</b> <a href="http://a301-6737-8537.sng.swissbank.com:5001/818">http://a301-6737-8537.sng.swissbank.com:5001/818</a>				
<b>Description:</b> An Ad HOC script which push the data from csv to Oracle database				

- Automation scripts written in Python, Shell etc. are all pushed to a GitHub Repository and the 'Link to source code' placeholder below the automation name redirects the page to the GitHub repository on click.
- Each automation has a unique ID in the catalog. Hence the links provided in the 'Access Automation via' field are unique and shareable between developers.

- The following screenshot is of the page when user clicks on the link with Catalog ID = 1.

http://a301-6737-8537.sng.swissbank.com:5001/

## Technology Operations Center - Automation Catalog

Search Automation..

[Home](#) [Register Your Automation](#) [User Guide<sup>4f</sup>](#) [Subscribe](#) [Contact Us](#)

### Browse Catalog

**By Source**

- ServiceNow
- Netcool

**By Type**

- Incident
- Run
- Query
- Request
- Governance and Op Design
- Package or Deploy
- Production Assurance
- Management Support
- Risk and Controls
- Continuous Improvement

**By Tools**

- IPCenter
- Automation Anywhere

#### Dynamic Correlation

IPCenter automatically correlates similar patterns raised in netcool within a specific timeframe, raising one SNOW ticket only and adding any future occurrences within the set

[Read more](#) ★★★★★



## *Routing Module*

All the routes are configured in the 'app-routing.module.ts' file. Following are the routes-

1. path = '', component = AutomationsComponent. This routes the application to the homepage.

URL= '<http://goto/toc-automation-catalog/>'.

2. path= '/register-automation', component = RegisterComponent. This routes to the registration page.

URL='<a href="http://goto/toc-automation-component/register-automation">http://goto/toc-automation-component/register-automation</a>'

3. path= '/subscribe', component = SubscribeComponent. This routes to the subscription page.

URL='<a href="http://goto/toc-automation-catalog/subscribe">http://goto/toc-automation-catalog/subscribe</a>'

4. path= '/contact-us', component = ContactusComponent. This routes to the contact-us page.

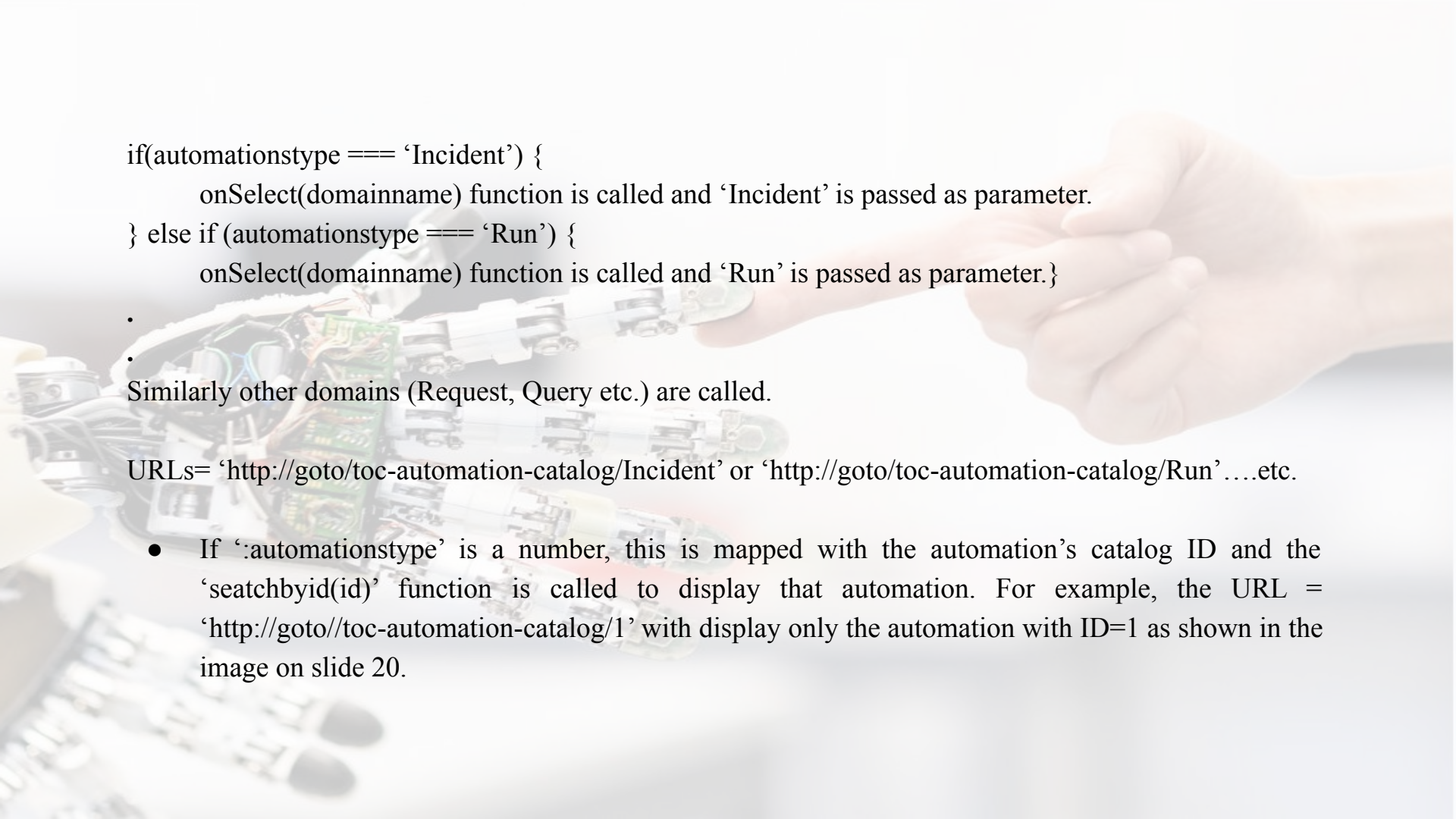
URL='<a href="http://goto/toc-automation-catalog/contact-us">http://goto/toc-automation-catalog/contact-us</a>'

5. path= ':automationstype', component = AutomationsComponent.

URL='<a href="http://goto/toc-automation-catalog/:automationstype">http://goto/toc-automation-catalog/:automationstype</a>'

'<a href="http://goto/toc-automation-catalog/:automationstype">:automationstype</a>' is a variable that is checked in the OnInit function of the Automations component to load the automations page.

It is matched in the following way-

A hand is pointing towards a circuit board. The background is a blurred image of a hand holding a circuit board.

```
if(automationstype === 'Incident') {  
    onSelect(domainname) function is called and 'Incident' is passed as parameter.  
} else if (automationstype === 'Run') {  
    onSelect(domainname) function is called and 'Run' is passed as parameter.}
```

- 
- 

Similarly other domains (Request, Query etc.) are called.

URLs= 'http://goto/toc-automation-catalog/Incident' or 'http://goto/toc-automation-catalog/Run'....etc.

- If ':automationstype' is a number, this is mapped with the automation's catalog ID and the 'seatchbyid(id)' function is called to display that automation. For example, the URL = 'http://goto//toc-automation-catalog/1' with display only the automation with ID=1 as shown in the image on slide 20.

## *Conclusion and Future Scope*

The catalog has seen a really quick spike in visitor count since its release (3000 hits in 3 days) and a good number of users have subscribed to the catalog.

### **Future Scope -**

- I. Automate addition of new automations into the database.
- II. Filter automations by execution stats, reusability considerations (high, medium and low).
- III. Further enhancement of search by introducing search by name, description, tags or all.

## ***Ticket Enrichment using Machine Learning***

This solution is designed for the support analysts that work on resolving the errors generated by different applications by providing 'labels' for the latest errors.

The labels best describe the errors that guide the analysts to know what steps to follow for their resolutions.

### **Objectives-**

1. To develop an automation in IPCenter that extracts the latest errors for particular jobs of an application by logging into the server and enriches the IT ticket data with the error labels predicted using Machine Learning.
2. To help incorporate the automation component with the wrapper automations developed by the automation developers of Asset Management and Investment Banking teams.



## *APIs used -*

Method	URL	Description
POST	/jff/job_onboard	Onboard a new job metadata to backend
POST	/jff/create_model	Create the Machine Learning model for a job in backend (please note the ML model for each job type needs to be created outside of this API by application SME)
POST	/jff/db_add_event	Insert each event data to batch failure database
GET	/jff/db_get_model	Retrieve the Machine Learning model detail from backend for the requested batch job
GET	/jff/predict_label	Trigger machine learning model and returns the predicted label for the error message
POST	/jff/db_add_enriched_event_data	Enriches the already inserted event data with the predicted label (this will be used for feedback mechanism with application SMEs in future)

- The following screenshots are of the IPCenter Automation Component

## Add event details (API add Event)

State Properties Extract Variables

Name: Add event details (API add Event)

State Type: NormalState(ID:5530621)

Action Type: HostCommandAction(ID:4202468)

Connection: pbjumpshost

Command:  
curl -X POST \${api\_add\_event} -H  
'Content-Type: application/json' -d  
'\${add\_eventdata}'

Secure STDOUT: ☐

Secure STDERR: ☐

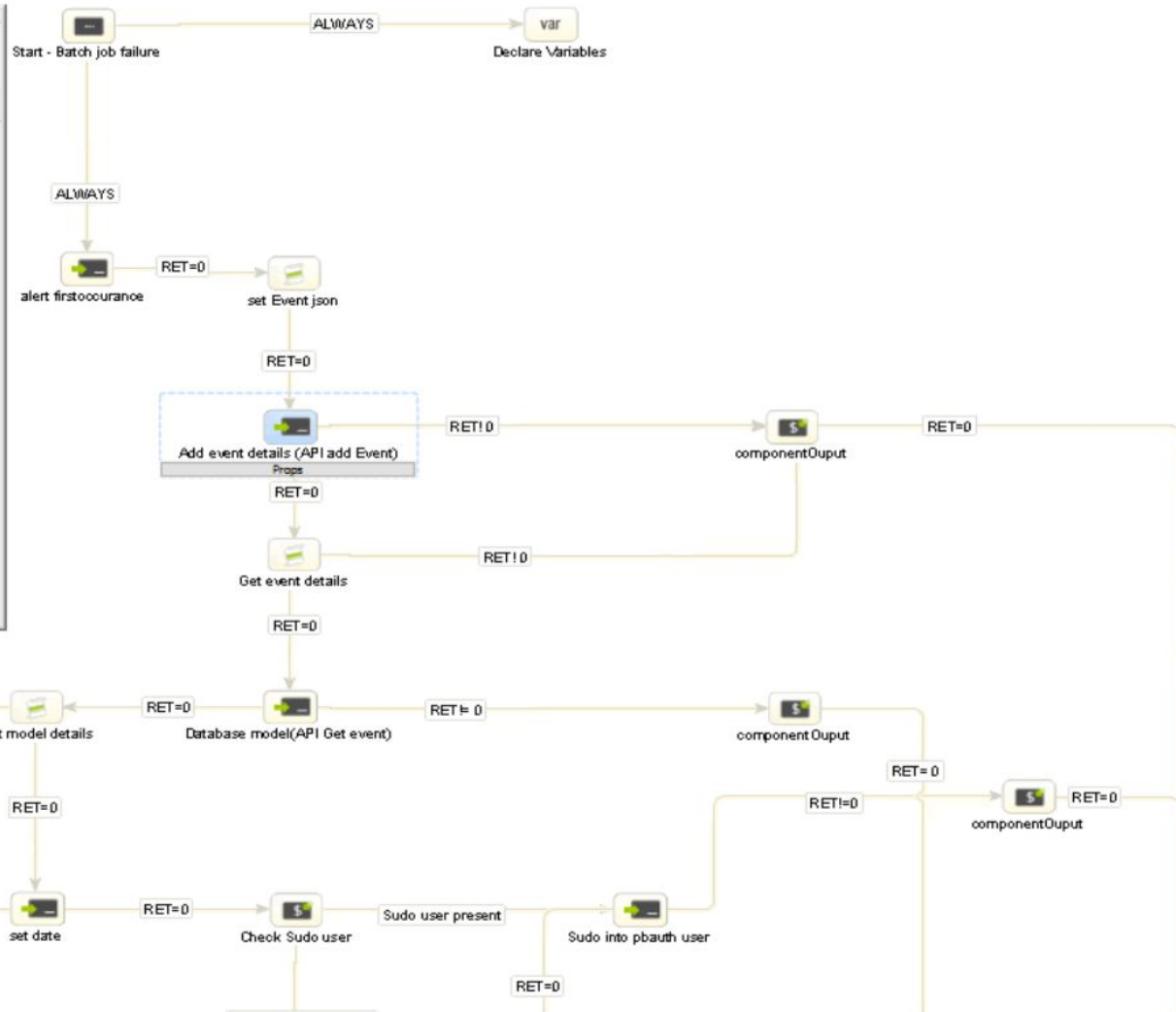
Auto-Trim: ☒

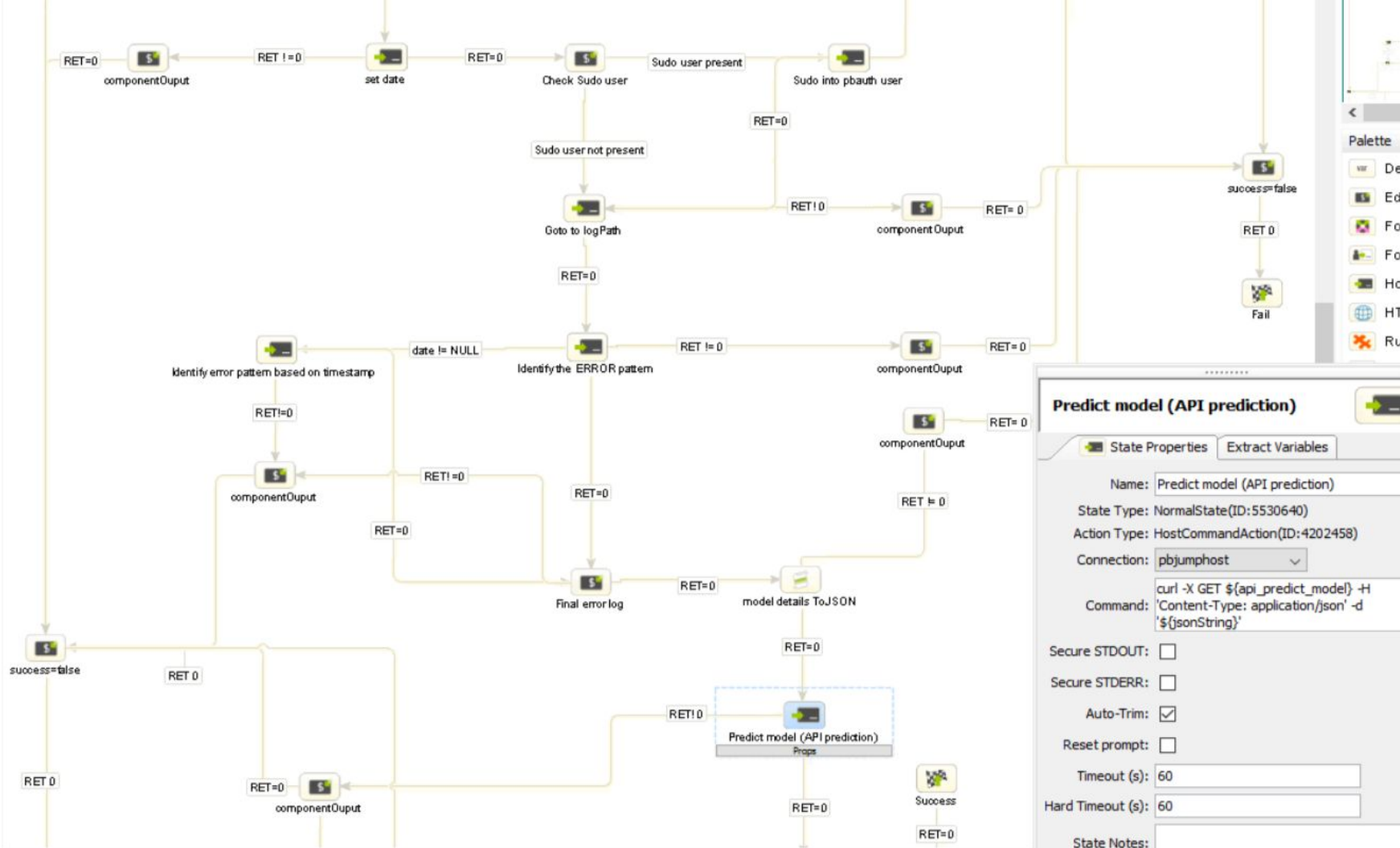
Reset prompt: ☐

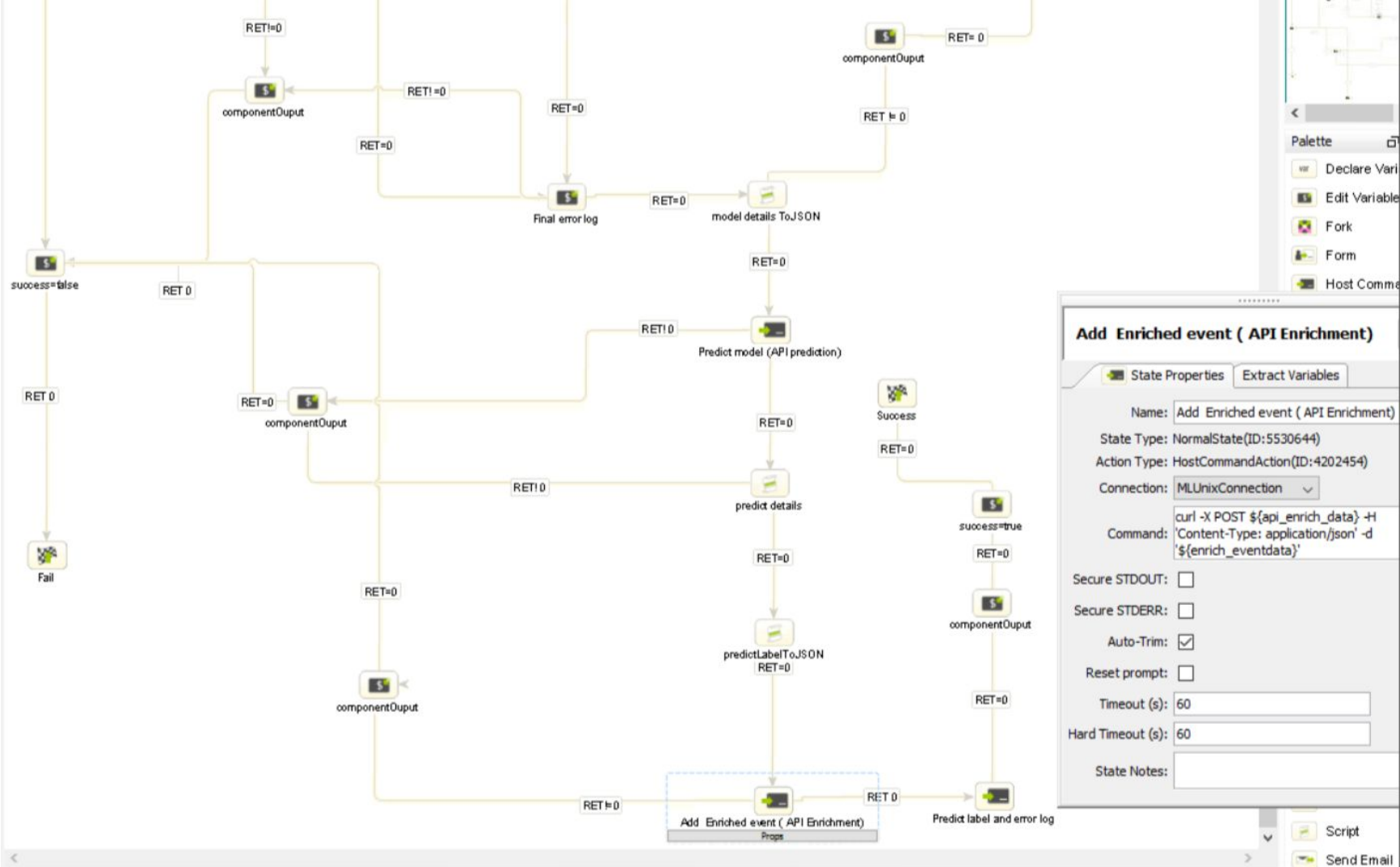
Timeout (s): 60

Hard Timeout (s): 60

State Notes:







## *Incorporating the IPCenter Automation Component*

The two teams - Asset Management and Investment Banking were assisted with incorporating the ML component into their respective IPC automation wrappers. Following are the steps followed for both the teams-

1. Job names and error log data (~90 days) of applications were collected from the teams.
2. Preprocessing of the errors data (removing headers and footers and manual labeling of the errors) was performed.
3. MLP algorithm was run on those error data.
4. The model was pickled and stored by calling the `create_model` API.
5. The ML automation component was added as a link in the automation wrappers developed by the respective teams.



## *Conclusion and Future Scope*

The ML automation component linked to the automation wrappers, enriched the tickets with the error labels.

The future scope is to automate some of the resolutions of the errors based on the error label predicted.

For example, most of the errors require a restart of the application. Hence, whenever such errors get predicted with the error label, the error label can be mapped with the process it must follow for resolution. Here, the process is restart of the application.



*THANK YOU!*