Introduction

The Pega Angular Starter Kit shows how to construct an Angular application to utilize the Pega-APIs with layout and field information.

If a business user changes the layout of section, number of fields or type of fields; then, with the information in the Pega-API, the Angular application should be able to adapt without re-coding.

The Angular Starter Kit example builds a Case Worker portal which supports *any* simple Pega application.

The CableConnect Sample Application is simple sample application that can be loaded on to your Pega Infinity 8.1.3 (and higher) system. This sample app exercises many of the Pega-API DX features.

Pega-API

In Pega Infinity, Pega-API's for cases/casetypes/assignments were updated with new extension points to provide layout/field information. This information comes from Harness/Section model information and can be used as hints to allow your Angular displays to be model driven.

This allows your displays to be linked to the Pega application, such that if the business user changes the design via the Case Designer or App Studio, your display can instantly reflect the changes, without re-coding.

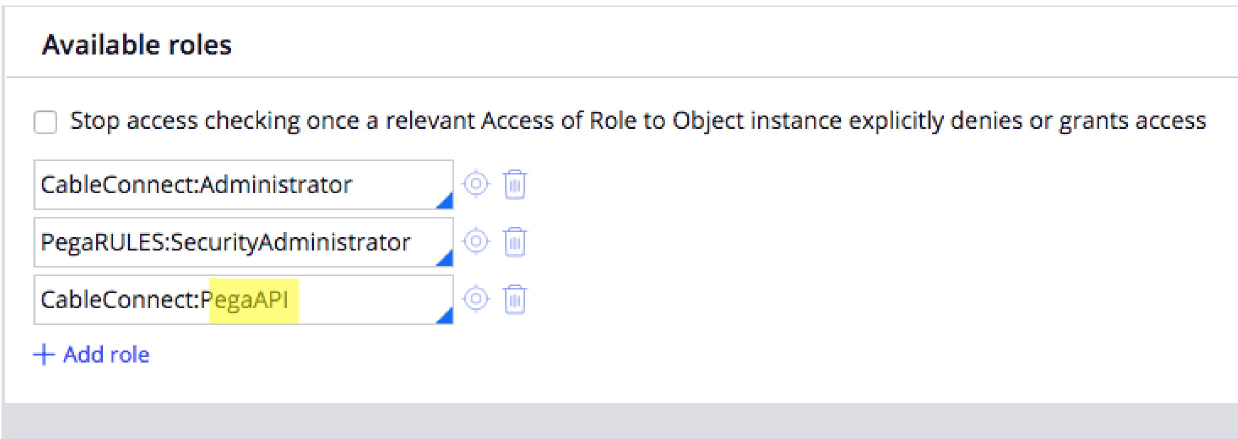
Layout and field information of a section/harness is passed through but organized differently to provide a simplified REST API JSON standard.

You can reference the Angular Starter Kit Installation Guide for more information.

Cable Connect Sample Application

In order to quickly understand the benefits and usage of the Angular Starter Kit, we have provided a sample Pega Infinity application called CableConnect which you can install on your Pega Infinity (8.1.3 and higher) system.

Note: If you create a new application or modify this application and create new access group, ensure that your access group contains the PegaAPI role, in order to interface with PegaAPI DX (the Pega Digital Experience APIs).



Description

Cable Connect is a simple application that consists of 3 users and admin:

- Representative – person contacted by a customer to get new service from a cable provider
- Tech – person who would fulfill the customer request and check off the request as being done
- Manager – person who can add a customer discount, if the representative requests one, and has access to multiple workbaskets.

Please note, that this application is very simple and not designed to be a full use case, but instead, be a simple application that you can use as an example.

Installation

- Download zip
- Login to Pega system
- Import (Configure>Application>Distribution>Import)
- Log off
- Log on as admin.cableco (see below)

Login

There are 4 logins for this application:

- rep.cableco, password: pega - case worker
- tech.cableco, password: pega – case worker
- manager.cableco, password: pega – case manager
- admin.cableco, password: pega – developer/admin

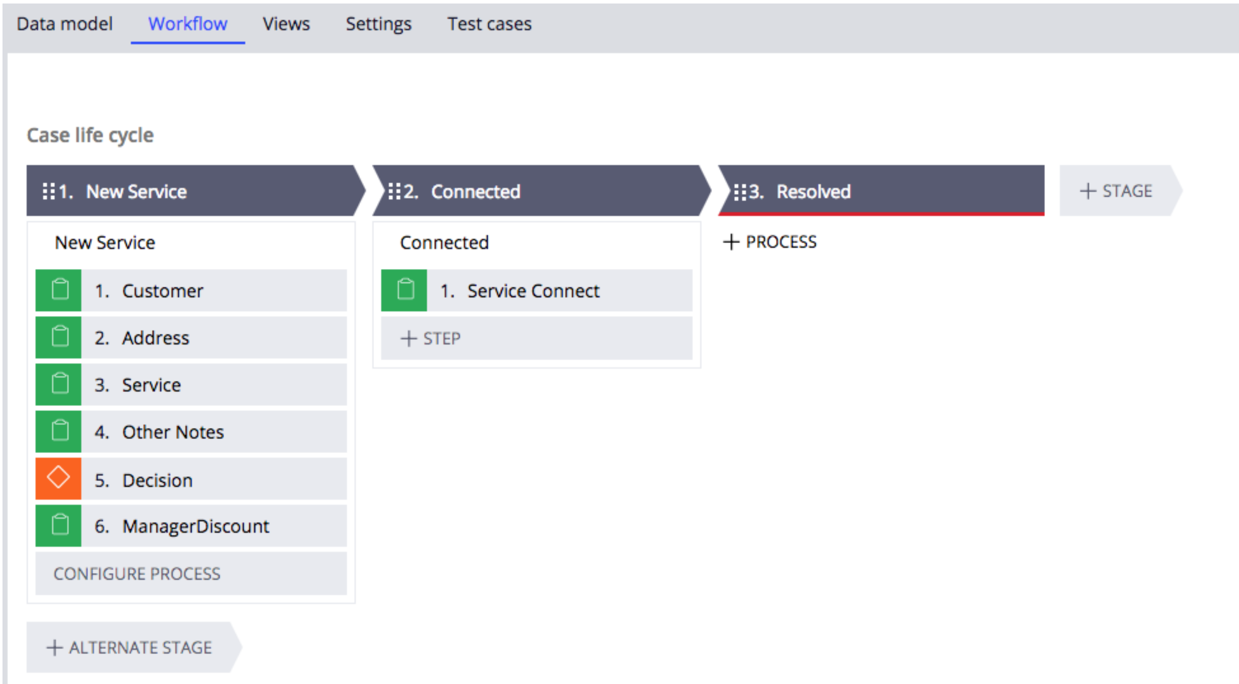
You can use these logins to update/edit/run the application. You might choose to run the application in your browser first, to understand the flow. See "step through" section for more information.

Supporting Functionality

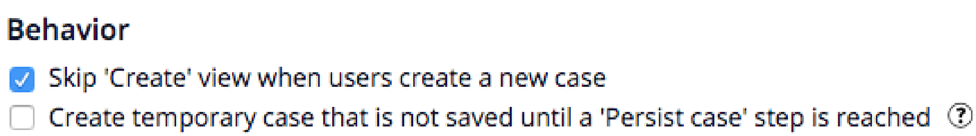
Cable Connect application supports the following functionality, that will be exercised in the Angular Starter Kit.

- Fields & Controls: pxTextInput, pxTextArea, pxCheckbox, pxRadioButtons, pxDropDown, pxAutoComplete, pxDateTime, pxButton, pxNumber & Disabled controls
- Modes: Editable, ReadOnly
- Layouts: Single Column, 3 Column
- Actions: Refresh with Post, Perform Action
- Validation: Property Edit Validate, Rule Edit Validate, Required
- Routing: Rep->Tech, Rep->Manager->Tech

Here is the Case life cycle.



Note: The majority of this application was created within the Case Designer.
Note: "Service" case type skips the "New" harness:



If you create an application that does not "skip" the "New" harness, then you WILL need to override the "New" harness in your application, as the out of the box "New" harness currently does NOT work with the PegaAPI DX.

The flow of this application is as follows:

rep.cableco

- gathers customer information, address, requested service and notes
- Rep determines whether to request a discount
- Rep is now finished – routed to Manager (if a discount is needed) or Tech to fulfill

manager.cableco

- Manager needs to get work items from manager workbasket
- If any work items request a discount, then manager can enter a discount

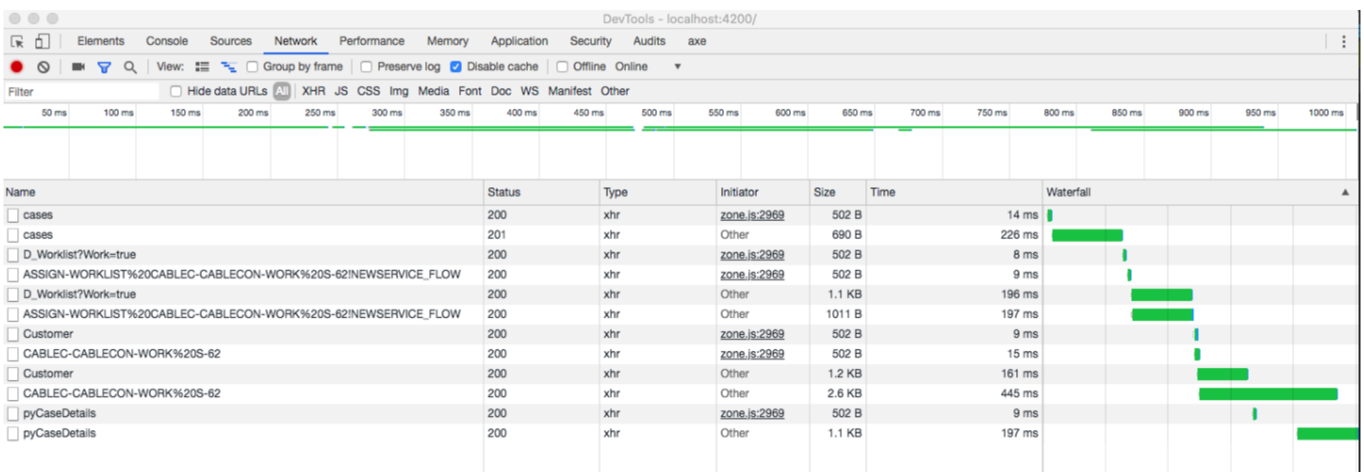
- Manager is now finished – routed to Tech to fulfill

tech.cableco

- Works is routed to Tech from either Rep or Manager (if discount)
- Tech is supposed to verify fulfillment of service
- Tech can edit "server request" and change if via "Update service"
- Tech resolves work item

A few points:

- This application is simple, so one can understand the basics.
- Work items get routed, so you can see multiple users and their corresponding work list and/or workbasket.
- Works items for manager get routed to a "manager workbasket" so you can see an example of work sent to a workbasket as well as see an example of how to show lists (display) of workbaskets based upon user's access group.
- Not all Pega API DX functionality and information is exercised here.
- Utilize the Chrome browser Developer Tools (Network tab) when running the Angular app so you can trace the network traffic and see the API calls, and responses.

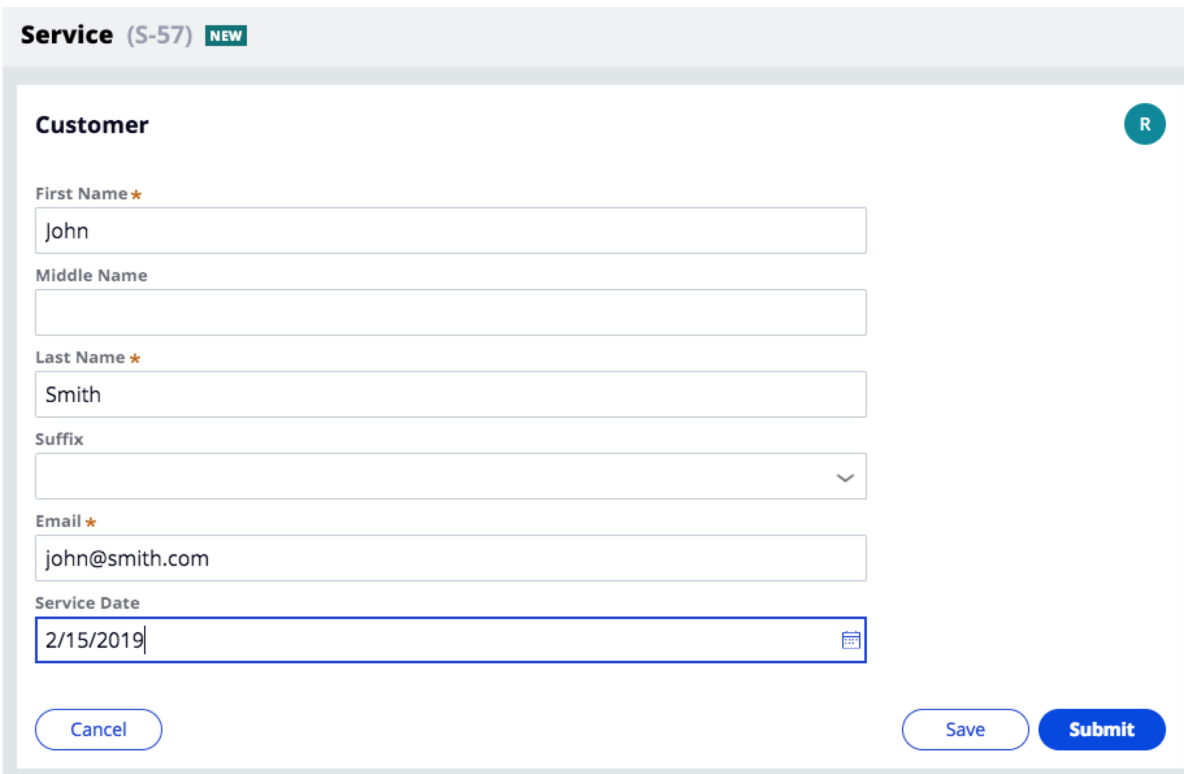


Step Through on the regular desktop

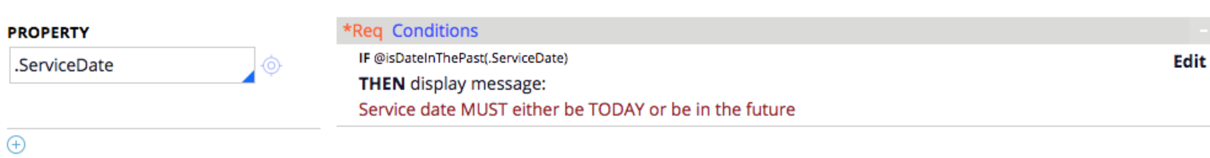
Customer

This is the first screen when logged in as "Rep". The following is exercised here:

- Text Input
- Required fields
- Autocomplete based upon Prompt List (Suffix)
- Email with Property Edit validate
- Date time with Rule Edit validation on Submit (date can't be in the past)



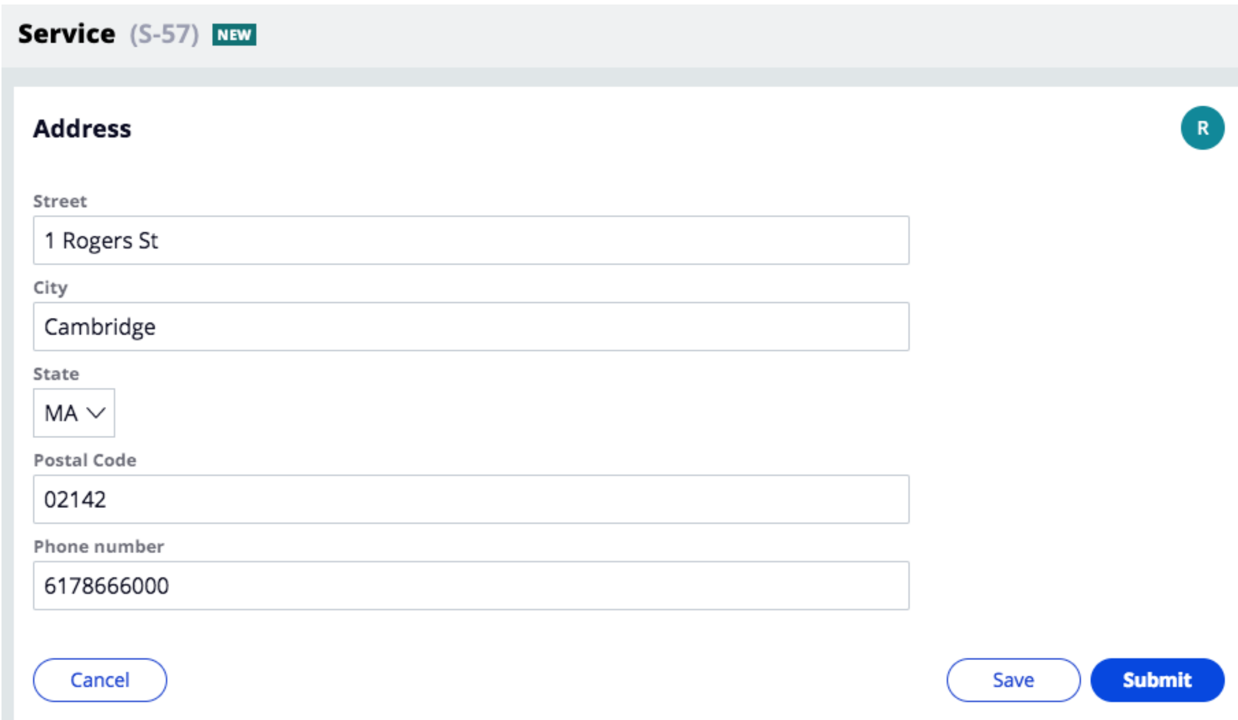
- Rule Edit Validate (ValidateCustomer)



Address

This is the second screen. The following is exercised here:

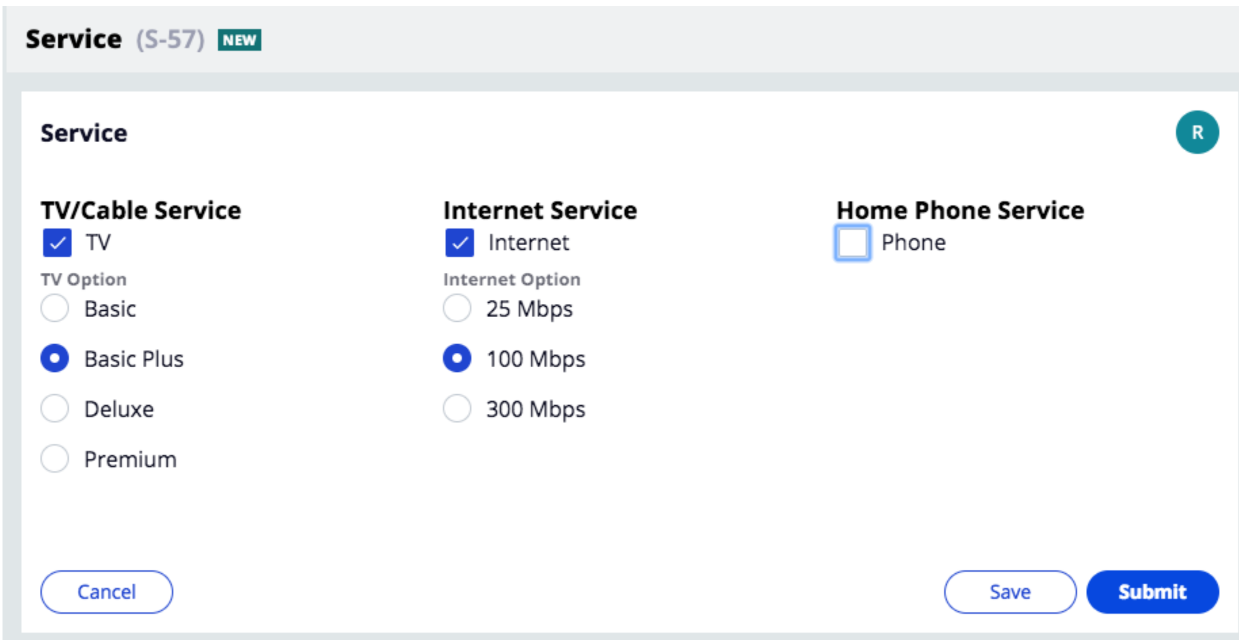
- Drop down with data based upon a data page (for State value)
- Phone number with Property edit validation (ValidPhoneNumber is overridden)



Service

This is the third screen. The following is exercised here:

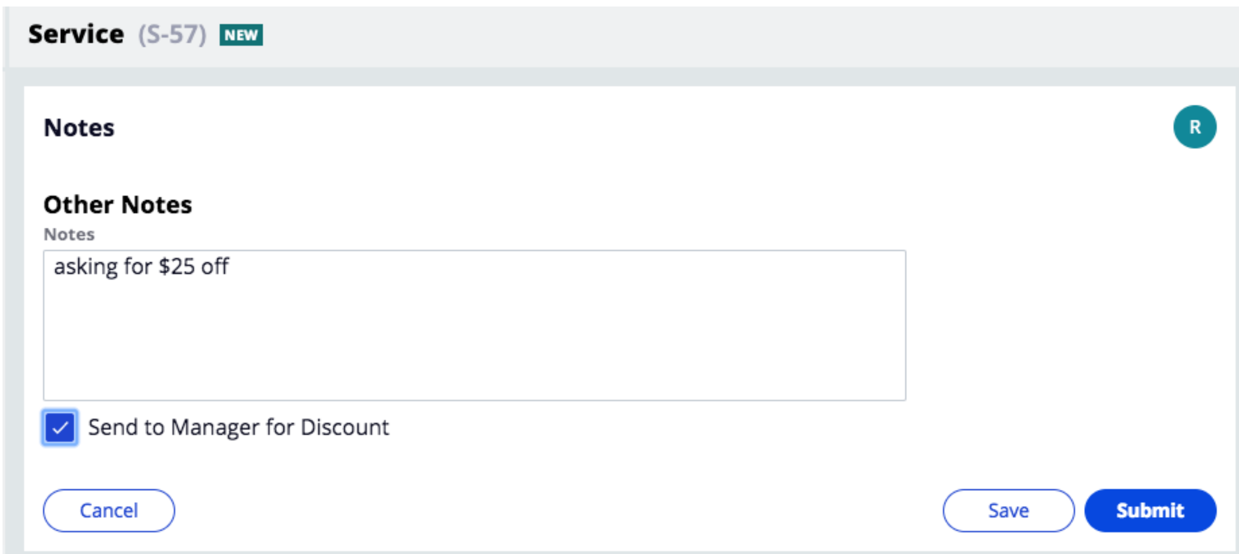
- Checkboxes have “Refresh” section. When checked, sub section appears
- Sub sections are visible via a Server side “when rule”
- Radio buttons



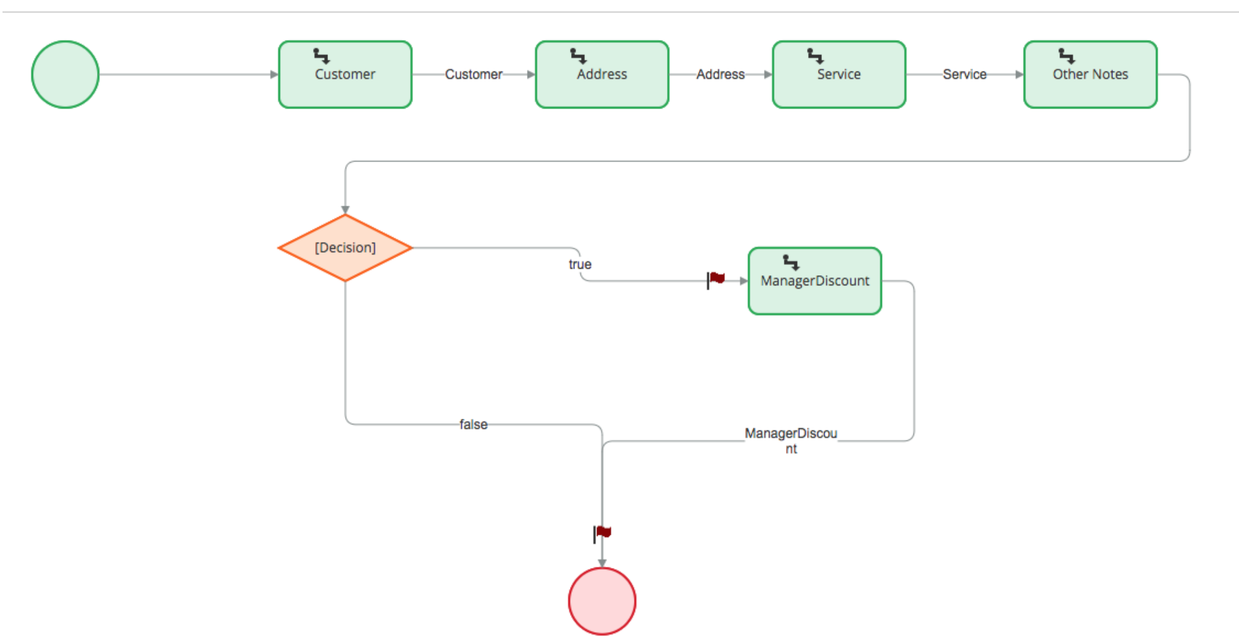
Notes

This is the fourth screen. The following is exercised here:

- If checkbox is checked, upon submit, work item will be routed to a manager workbasket. Otherwise, will be routed to "Tech" user.



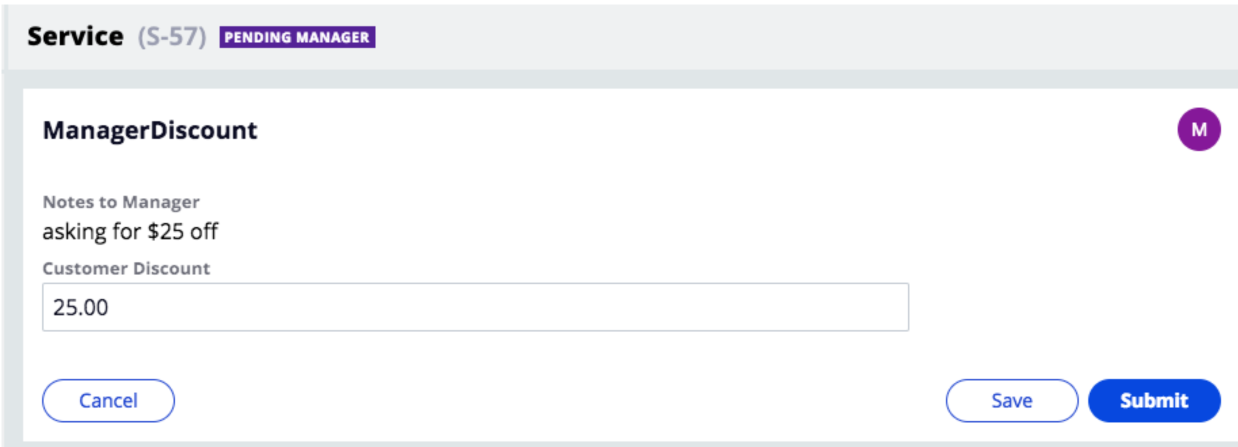
In the flow, we have a decision based upon the checkbox



Manager Discount

This screen is seen by the manager. Work item will appear in manager worklist. Manager can add a customer discount. The following is exercised here:

- ReadOnly
- Number

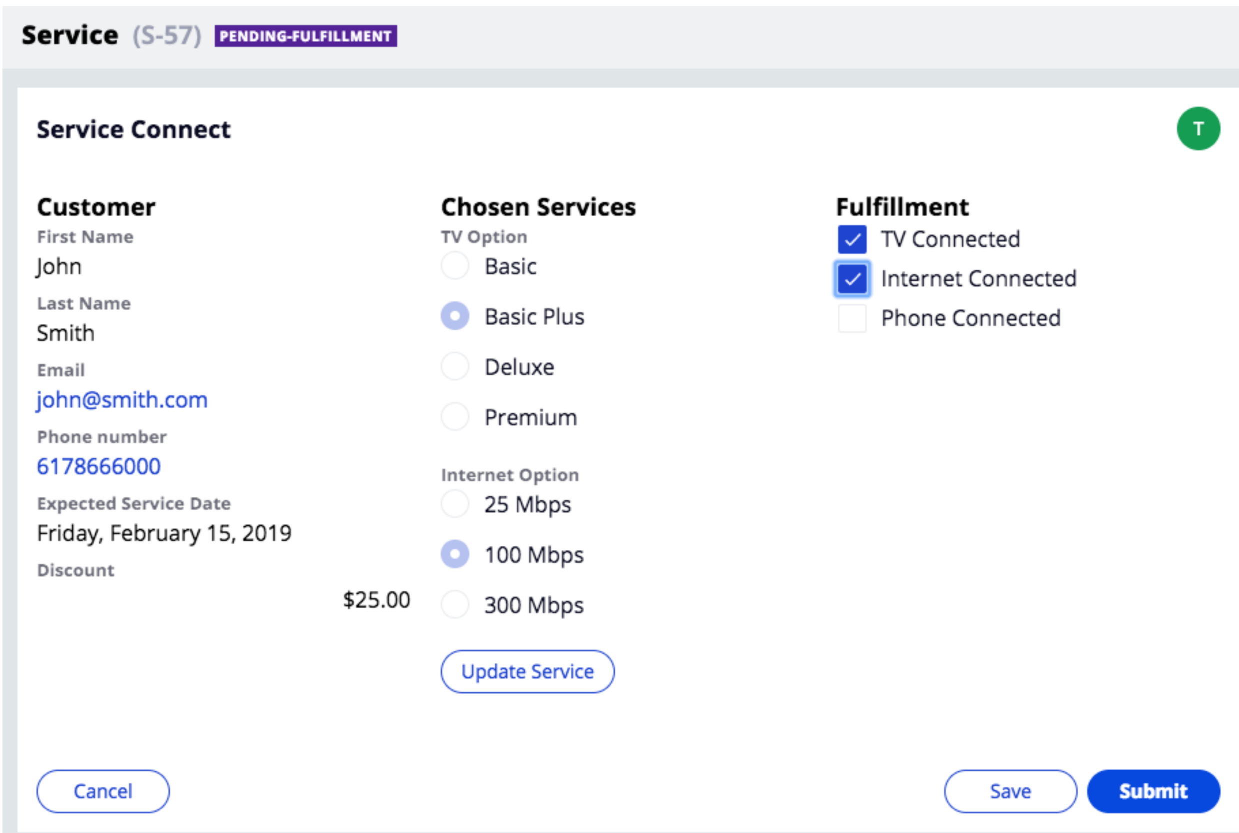


Service Connect

This screen is seen by the "Tech" user. The "Tech" user is expected to fulfill the request and check off that which has been done. The "Tech" user has the ability to update the "service" (see update service)

The following is exercised here:

- Display Text (formatted text)
- Disabled fields
- Perform Action (call local action to update server) – Update Service

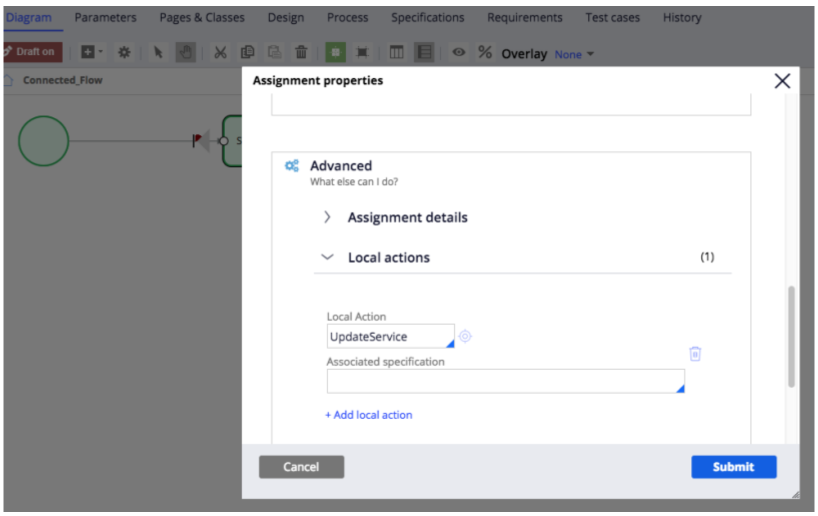


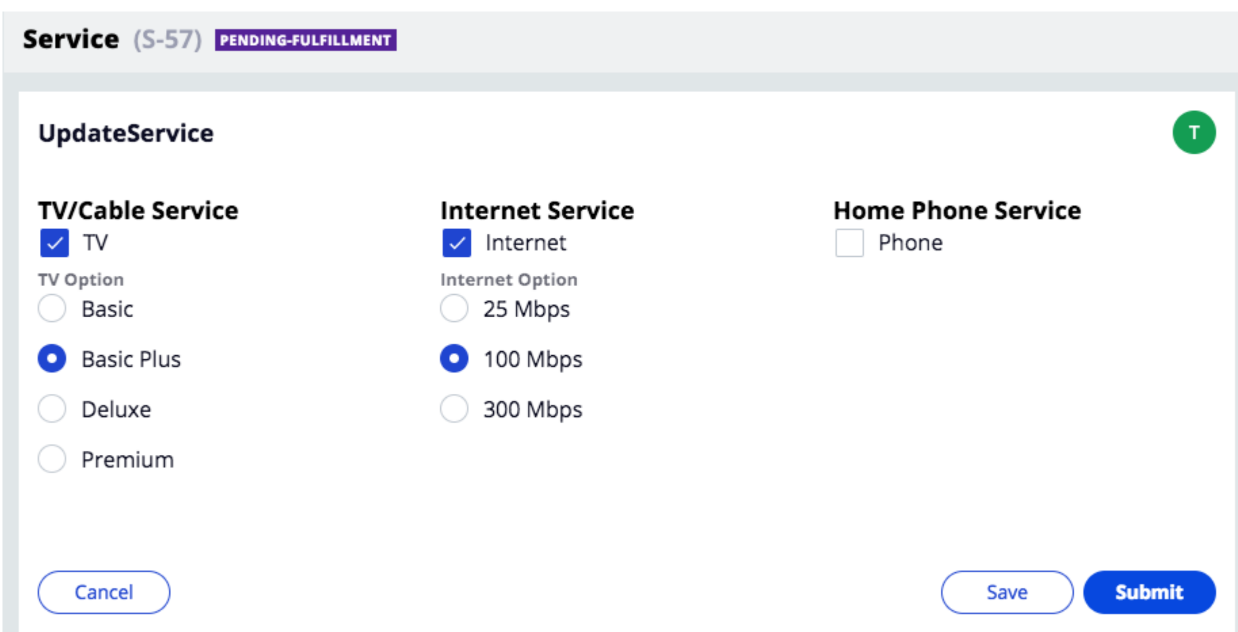
Update Service

This screen is seen if “Update Service” button is pressed in prior screen. This is a local action that can update the “service”. Once the change of service is made, submit will be pressed and the work item will be routed back to the “ServiceConnect” screen.

The following is exercised here:

- Local action on an assignment



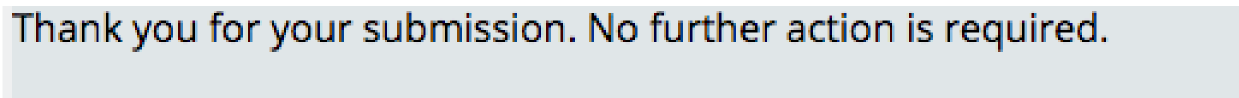


Confirm

During the process, the user can see a confirm screen when no further action is required. The Confirm harness has to be overridden, as the default will currently not work with PegaAPI DX.

The following is exercised here:

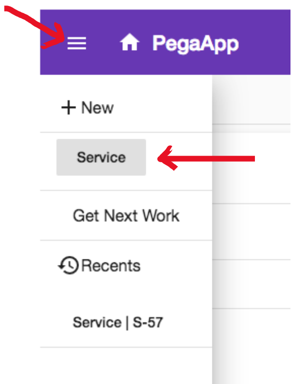
- Overridden Confirm harness with screen layout



Step Through Angular Application

After login

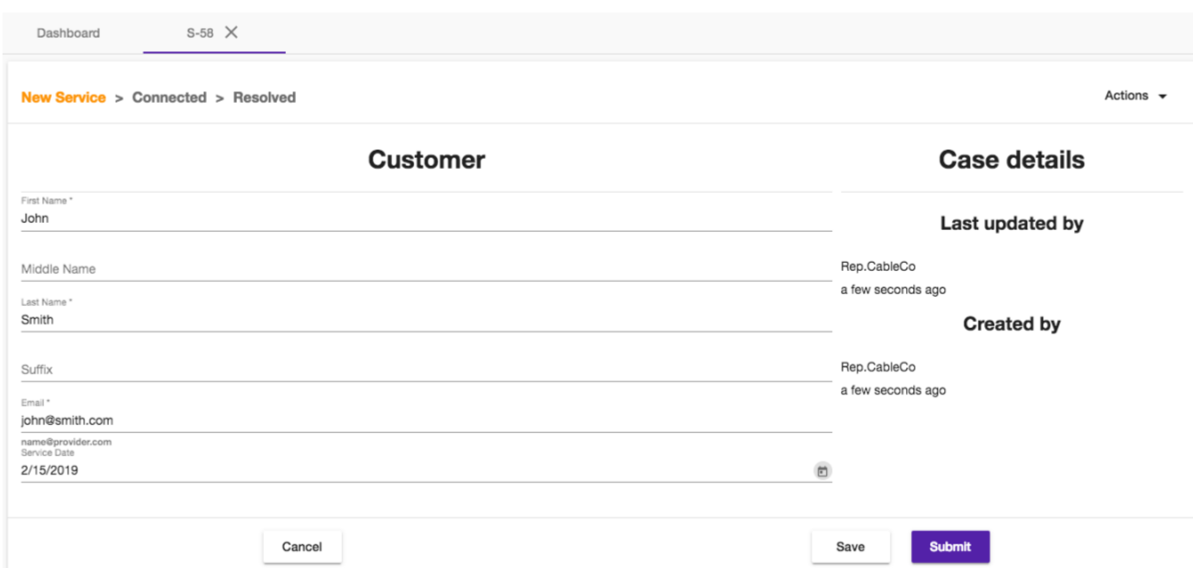
Create a new Service work item, select the hamburger icon, then select “Service”



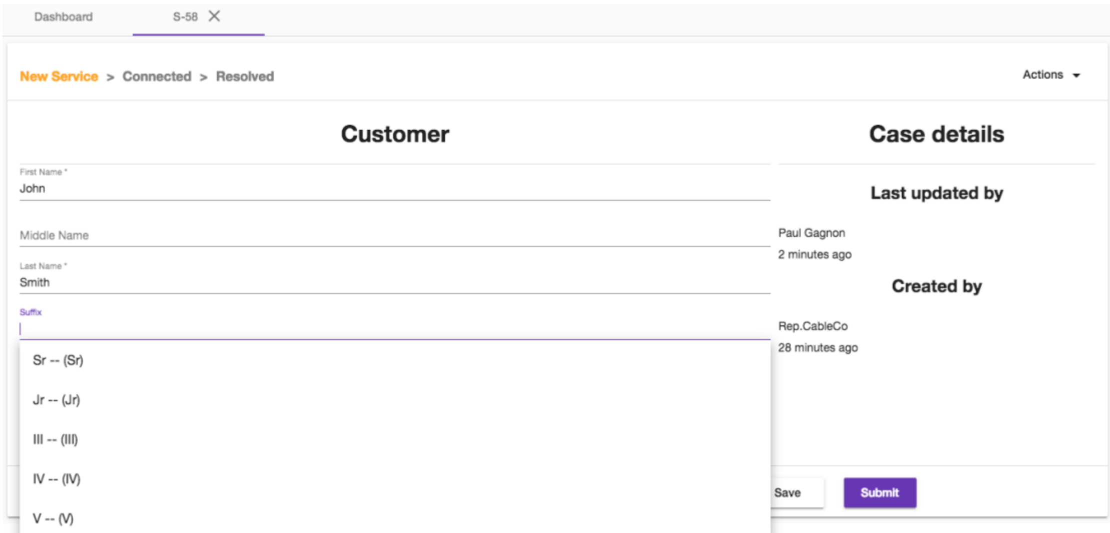
Customer

This is the first screen when logged in as “Rep”. The following is exercised here:

- Text Input
- Required fields, error handling
- Autocomplete based upon Prompt List (Suffix)
- Tooltips (Angular hints)



- Autocomplete via prompt list



Error handling

- Required – local error handling in Angular App

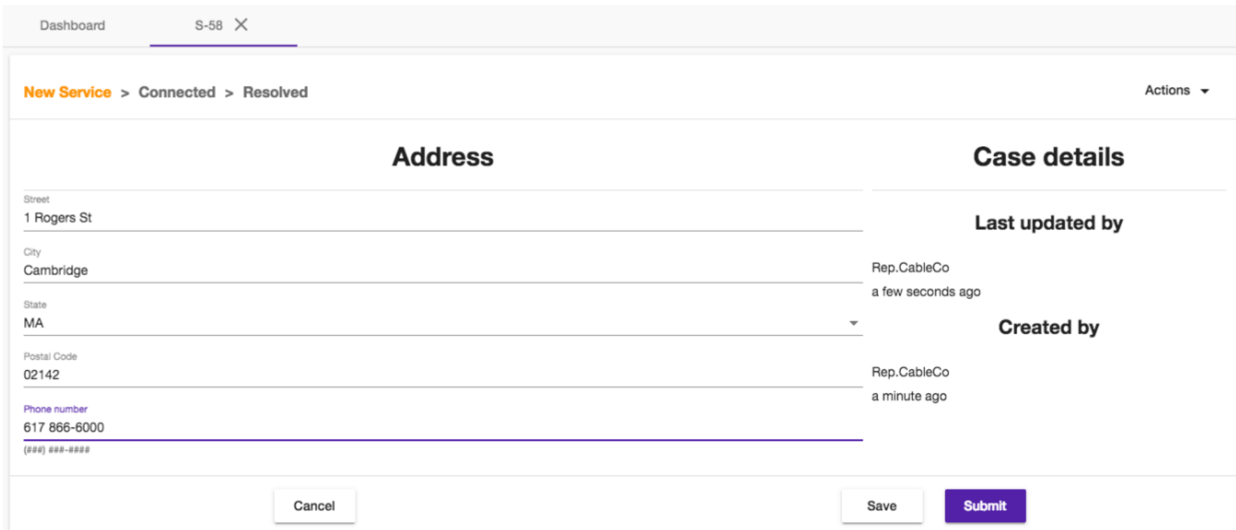
- Bad email – both local in Angular App, and on the server via a submit

- Bad Date (not in the future) – on server via submit

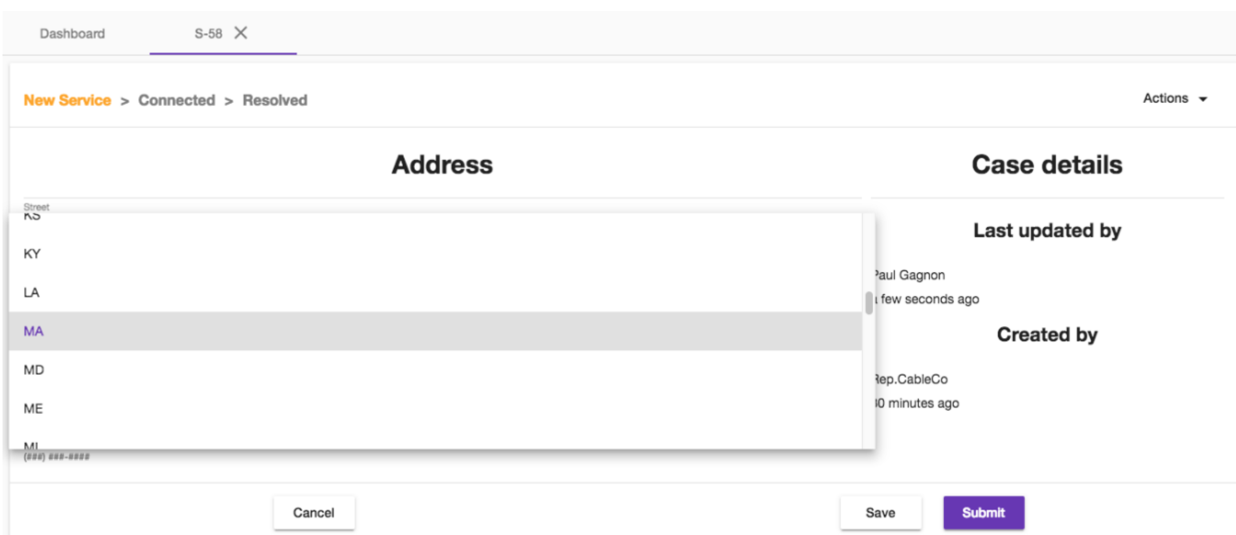
Address

This is the second screen. The following is exercised here:

- Drop down with data based upon a data page (State)
- Phone number with Property edit validation (default is overridden)



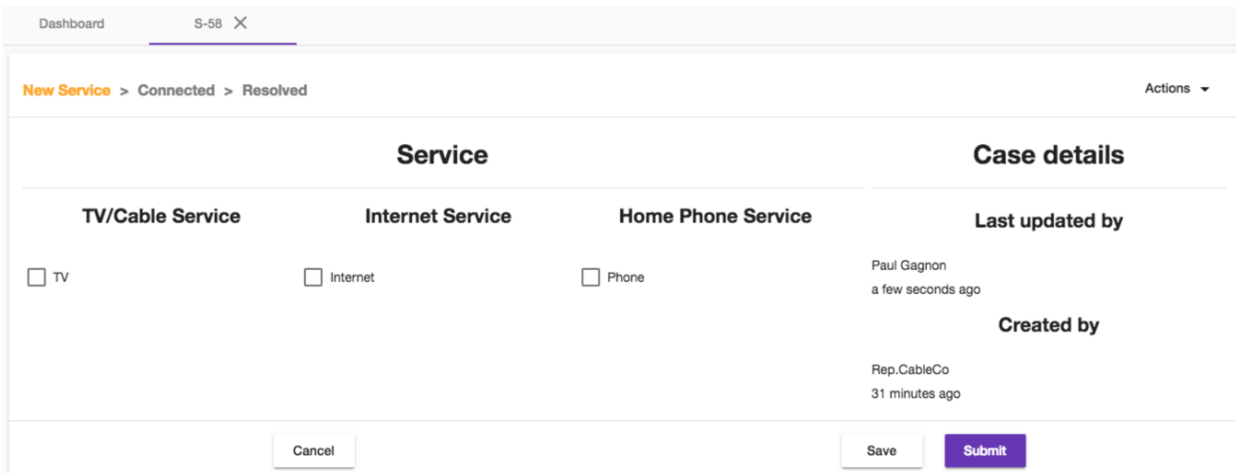
- Drop Down via data page



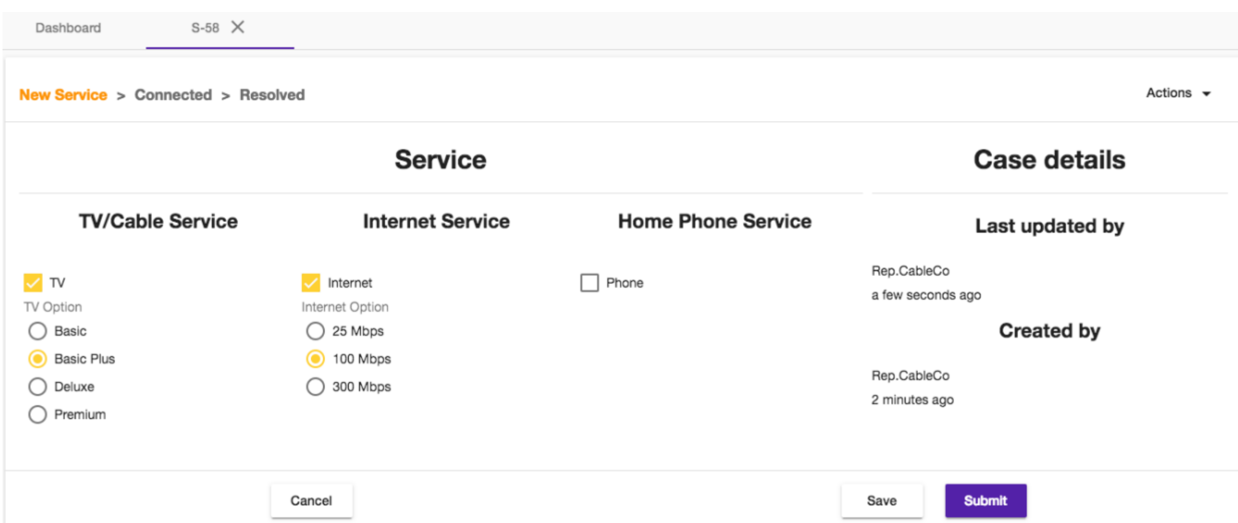
Service

This is the third screen. The following is exercised here:

- Checkboxes have "Refresh" section. When checked, the sub section appears
- Sub sections are visible via a Server side "when rule", so when you call the "refresh" API, the server will recalculate based upon "when rules" and return data that will have the new section(s).
- Radio buttons



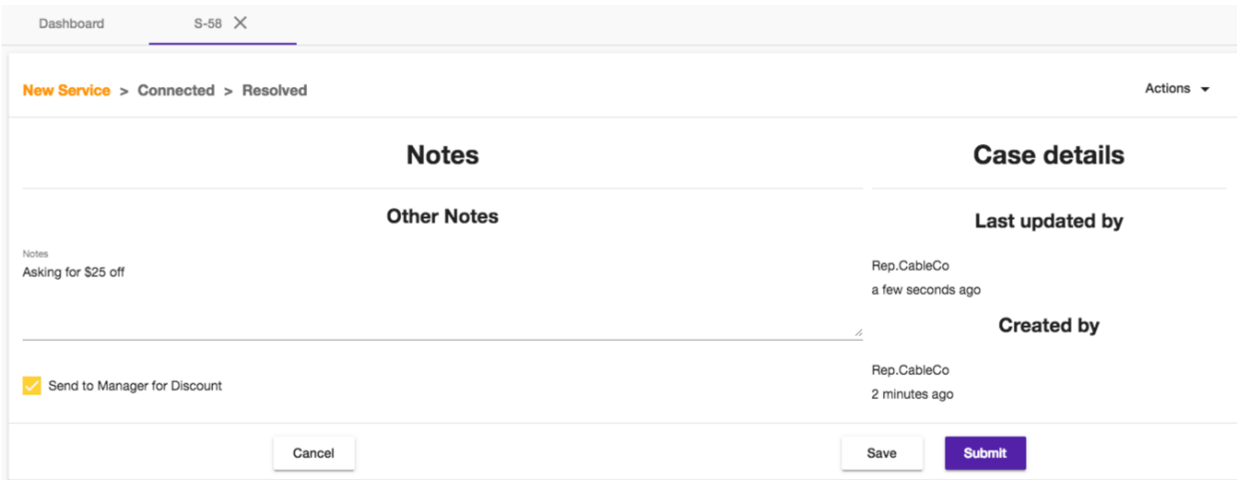
- Selecting checkboxes causes refresh and visible sub sections



Notes

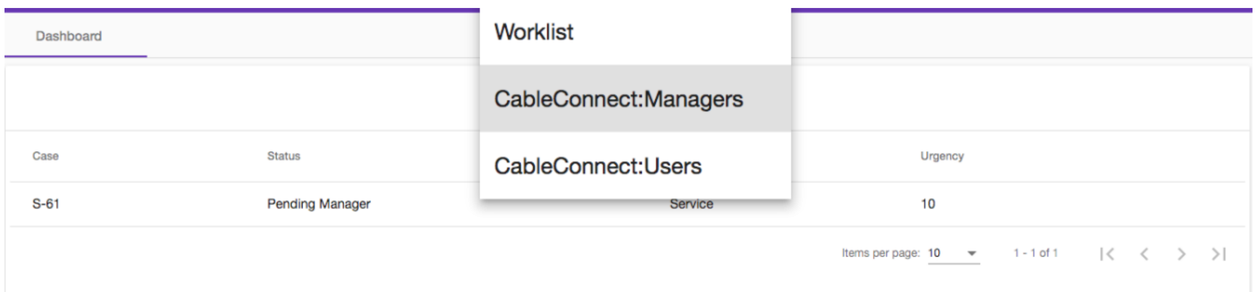
This is the fourth screen. The following is exercised here:

- If checkbox is checked, upon submit, work item will be routed to a manager workbasket. Otherwise, will be routed to "Tech" user.



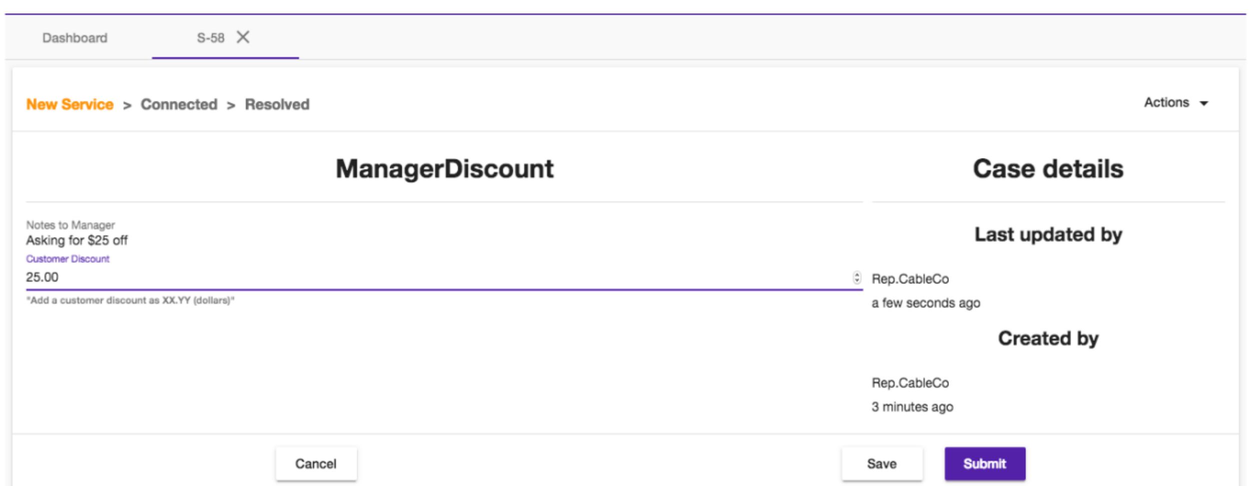
Manager Discount

This screen is seen by the manager. Work item will appear in manager worklist.



Manager can add a customer discount. The following is exercised here:

- ReadOnly field
- Number (only numbers can be entered)

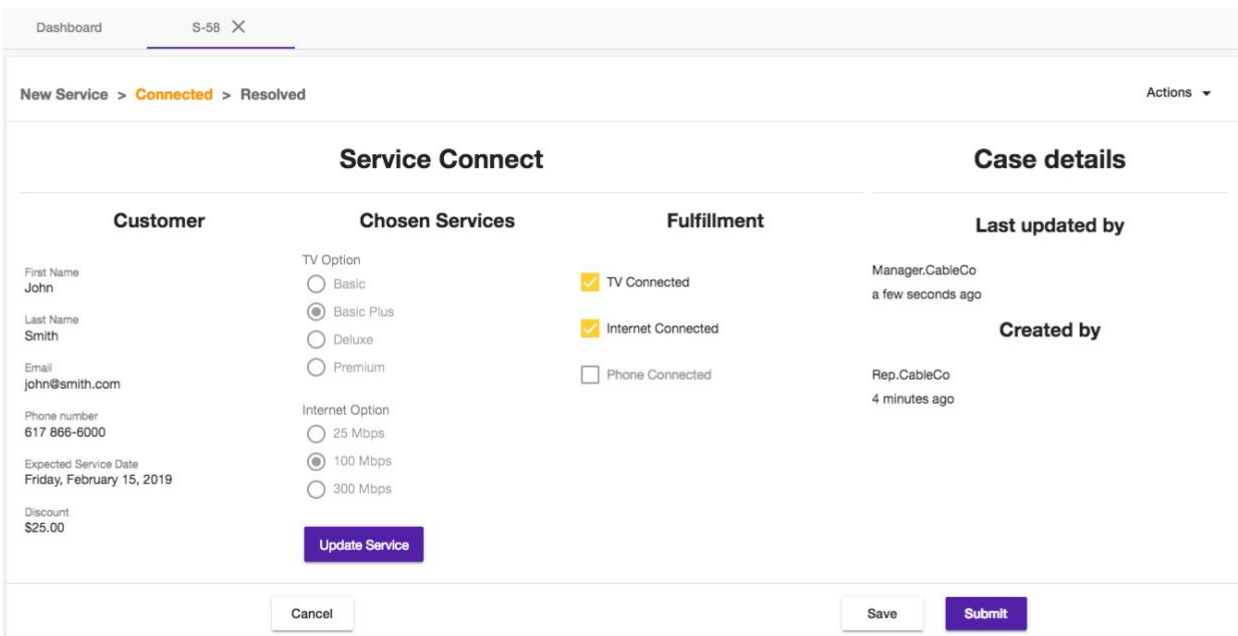


Service Connect

This screen is seen by the “Tech” user. The “Tech” user is expected to fulfill the request and check off that which has been done. The “Tech” user has the ability to update the “service” (see update service)

The following is exercised here:

- Display Text (formatted text, email, date, currency)
- Disabled fields
- Perform Action (call local action to update server)
- Triple Layout

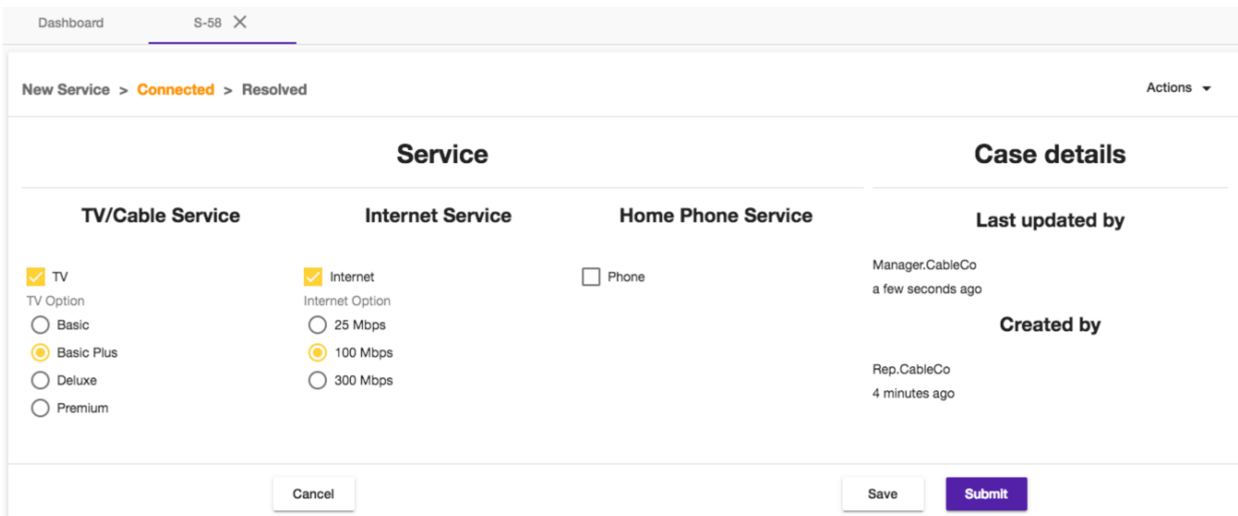


Update Service

This screen is seen if "Update Service" is pressed in prior screen. This is a local action that can update the "service". After submit, the work item will be routed back to the "ServiceConnect" screen.

The following is exercised here:

- Local action on an assignment



Confirm

During the process, the user can see a confirm screen when no further action is required. The Confirm harness has to be overridden, as the default will currently not work with PegaAPI DX (Digital Experience APIs).

The following is exercised here:

- Overridden Confirm harness with screen layout