

PEGASYSTEMS PROVIDES THIS SOFTWARE 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT. IN NO EVENT WILL PEGASYSTEMS, THE AUTHORS, OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH THIS SOFTWARE OR THE USE OR OTHER DEALINGS IN THIS SOFTWARE.

# Introduction

Pega's Angular Starter Pack shows how to construct an Angular application to utilize Pega's [DX API V1](#) with layout and field information.

If a business user changes the layout of a section, the number of fields, or the type of fields; then, with the information in the DX API, the Angular application adapts to them automatically.

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

The CableConnect Sample Application is a sample application that you can load into your Pega Infinity 8.3 (and later releases) system. This sample app uses several DX API features.

## DX API

Pega APIs augmented with UI metadata response information are DX APIs. In Pega Infinity, select APIs for instances cases/casetypes/assignments are 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.

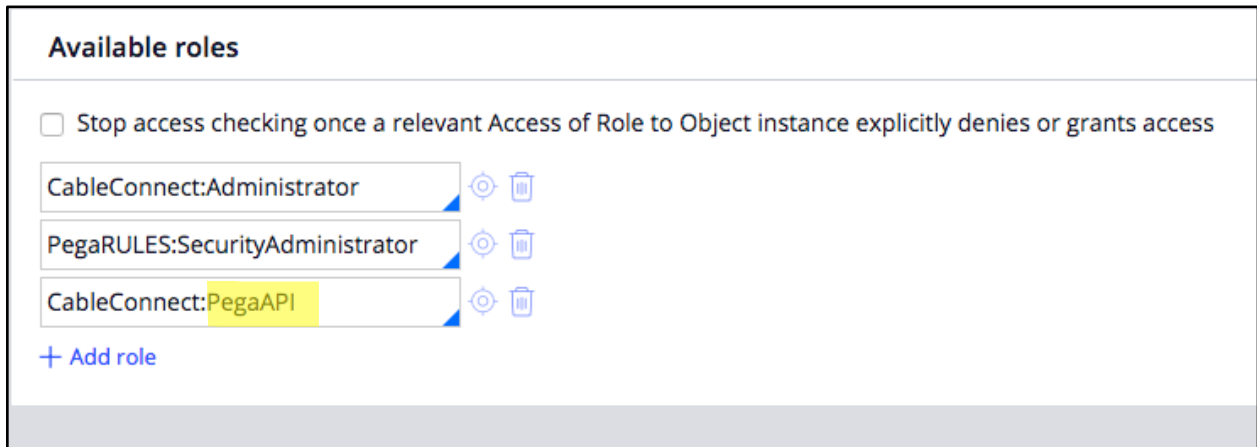
Layout and field information of a section/harness is passed through but organized differently to provide a simplified REST API JSON standard.

For more information, see the Angular Starter Pack's Installation Guide.

# CableConnect Sample Application







A sample application, CableConnect, is included in the starter pack to enable you to quickly understand the benefits and usage of the Angular Starter Pack. Import the application into your Pega Infinity (8.3 and later) system to use it.

Note: If you create a new application or modify this application and create a new access group, ensure that your access group contains the PegaAPI role. This is to ensure interface with DX API.



**Available roles**

☐ Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access

CableConnect:Administrator		
PegaRULES:SecurityAdministrator		
CableConnect:PegaAPI		

[+ Add role](#)

## Description

CableConnect represents a typical service request scenario in a cable company where a customer contacts a case worker for service. The case worker raises a service request which is then routed to the case manager or a technical case worker for further processing. The technical case worker uses the application to update the status of the service request and resolve the case.

The application consists of three types of users and an administrator.

- Rep: A user contacted by a customer for service.
- Tech: A user who fulfills the customer request and then marks the request as fulfilled.
- Manager: A user who adds a customer discount, if the representative requests one, and has access to multiple workbaskets.

Each user type is associated with an operator. For more information, see the [Roles and operators](#) section. For additional information about the CableConnect application, see [CableConnect \(sample application\)](#).

Note: The application's design enables you to explore the capabilities of the starter pack. Use this application as an example only, not a full-fledged case.

## Roles and operators

There are four operator logins for the CableConnect application:

- customer.cableco, password: pega – case worker (Customer)
- rep.cableco, password: pega - case worker (Representative)
- tech.cableco, password: pega – case worker (Tech)
- manager.cableco, password: pega – case manager (Manager)
- admin.cableco, password: pega – developer/admin (Administrator)

Use the operator logins to update/edit/run the application. You might choose to run the application in your browser first, to understand the flow. For more information, see the [Step Through on the regular desktop](#) and [Step Through Angular Application](#) sections.

## Installation

Import the CableConnect application into Pega Platform to use it. For detailed information about the import procedure, see [Importing CableConnect](#).

A gist of the installation procedure is as follows:

1. Download the starter pack .ZIP file.
2. Extract the .ZIP file's contents.
3. Log into Dev Studio.
4. Import the application .ZIP file (Configure>Application>Distribution>Import).
5. Log off.
6. Log in as admin.cableco.

### OAuth 2.0

CableConnect supports OAuth2.0 in the Angular starter pack. For more information, see the **AngularApplication.PDF** file in the starter pack.

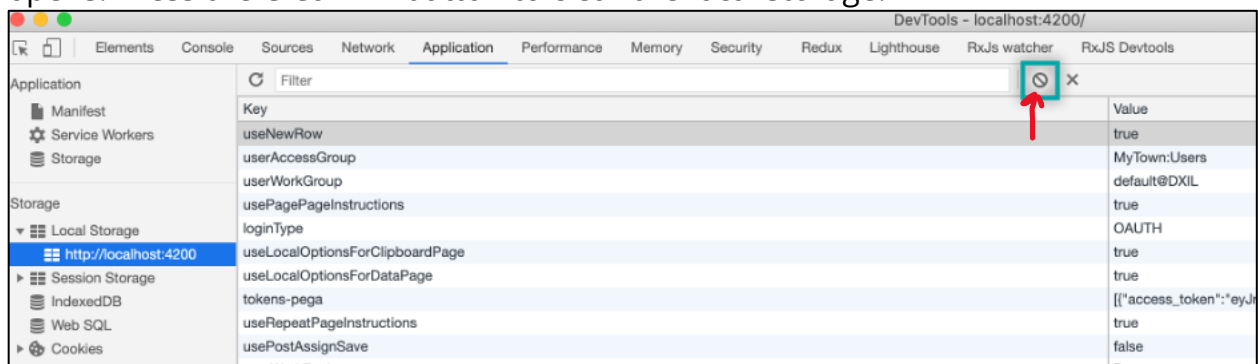
Basic configuration to enable OAuth 2.0:

1. Set the API service package to OAuth 2.0
2. Enter the Client ID from OAuth 2.0 Client Registration *CableCoV1OAuth* and a correct redirect URI. Here, the Client ID is *6203101846007304421* and the default redirect URI is *http://localhost:4200/*.
3. Select OAuth on the Angular login page and enter Client ID.
4. The screen redirects to the Pega Infinity login screen, enter the login credentials. The screen then redirects back to Angular and you see the dashboard.
5. Log off.

NOTE: First-time login sometimes takes a while.

With OAuth 2.0 and this starter pack, we recommend that you log off before killing the browser or terminating the session. The starter pack clears the local storage of tokens used during the redirect. Otherwise, you may have problems with logins to different applications and/or users.

To troubleshoot login-related issues, clear the local storage of the browser. In Chrome, open the Chrome debugger and click the **Application** tab, select **Local Storage**, and then select the server. The selected server's local storage information opens. Press the **Clear All** button to clear the local storage.



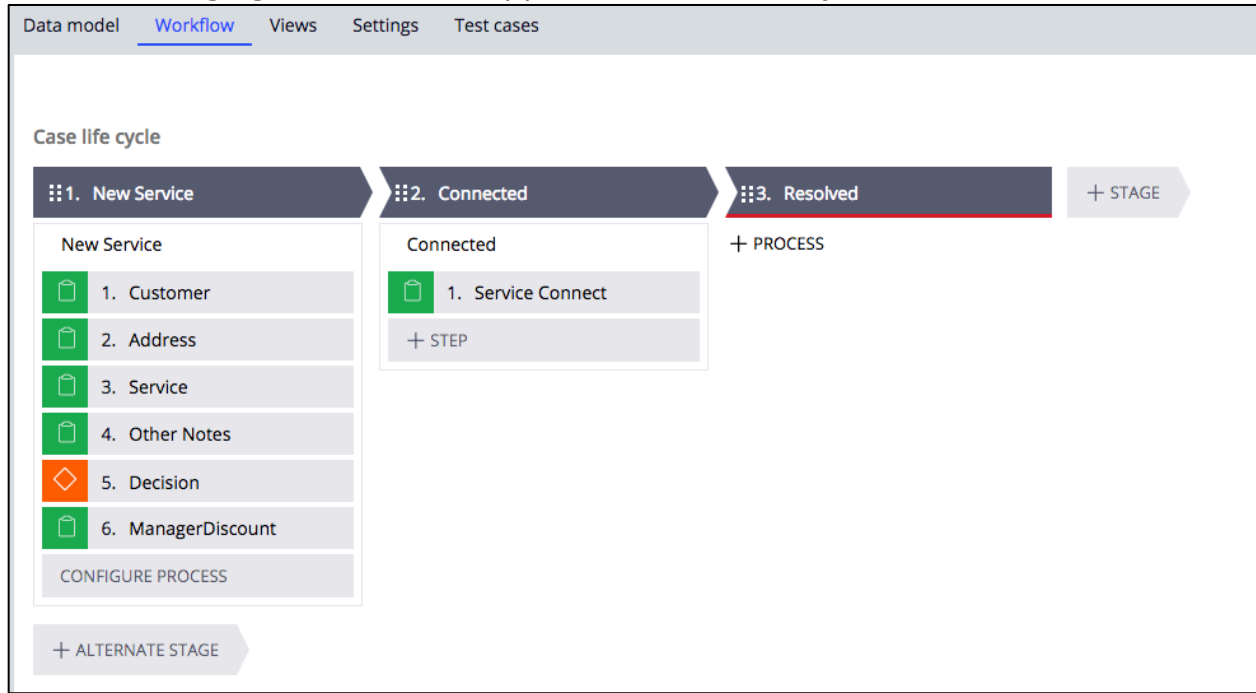
## Supporting Functionality

The CableConnect application supports the following functionality:

- Fields & Controls: *pxTextInput*, *pxTextArea*, *pxCheckbox*, *pxRadioButtons*, *pxDropDown*, *pxAutoComplete*, *pxDateTime*, *pxButton*, *pxNumber* & Disabled controls
- Modes: Editable and Read-only
- Layouts: Single column and three Column
- Actions: Refresh with Post and Perform Action

- Validation: Property Edit Validate, Rule Edit Validate, and Required
- Routing: Rep→Tech, Rep→Manager→Tech

• The following figure shows the application's case life cycle:



Most of the application has been created using Case Designer. The Service case type's behavior has been designed to skip the New harness:

**Behavior**

☒ Skip 'Create' view when users create a new case

☐ Create temporary case that is not saved until a 'Persist case' step is reached ?

If you create an application that does not skip the New harness, then you must override the New harness in your application, as this feature does not work with DX API out of the box, currently.

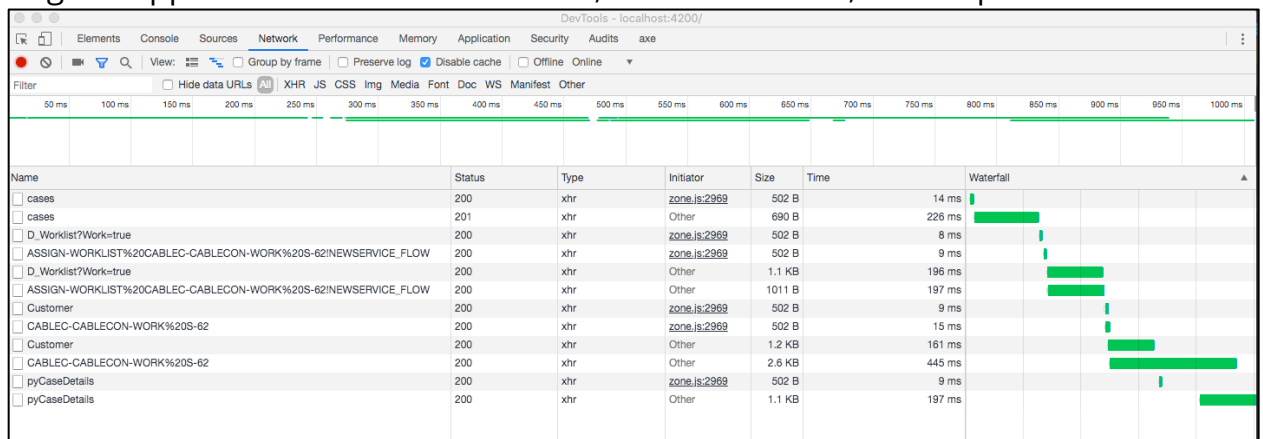
The flow of this application is as follows:

- rep.cableco
  - Gathers customer information, address, requested service, and notes.
  - Determines whether to request a discount.
  - Submits the service request, which is then routed to Manager (if a discount is needed) or Tech to fulfill by the application.

- manager.cableco
  - Fetches work items from the Manager workbasket.
  - Adds a discount if any work items request a discount.
  - Submits the service request, which is then routed to Tech for service fulfillment.
- tech.cableco
  - Receives service requests routed by the application to Tech from either Rep or Manager.
  - Verifies service fulfillment and updates the service request.
  - Optionally, modifies the service request.
  - Resolves the service request.

A few points to consider:

- The CableConnect application is simple, so one can understand the basics.
- The application routes work items, so you can see multiple users and their corresponding work list and/or workbasket.
- The application routes work items for Manager 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 the user's access group.
- Not all DX API functionality and information is implemented here.
- Use the Chrome browser Developer Tools (Network tab) when running the Angular app to trace the network traffic, see the API calls, and responses.



# Step Through on the regular desktop

## Customer

Customer is the first screen when logged in as Rep. This screen uses the following UI components:

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

**Service (S-57) NEW**

**Customer**

First Name ★  
John

Middle Name

Last Name ★  
Smith

Suffix

Email ★  
john@smith.com

Service Date  
2/15/2019

Cancel Save Submit

- Rule Edit Validate (ValidateCustomer)

**PROPERTY**

.ServiceDate

**\*Req Conditions**

IF @isDateInThePast(.ServiceDate)  
**THEN** display message:  
Service date MUST either be TODAY or be in the future

Edit

## Address

Address is the second screen. This screen uses the following UI components:

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

**Service (S-57)** NEW

**Address** R

Street

1 Rogers St

City

Cambridge

State

MA ▾

Postal Code

02142

Phone number

6178666000

Cancel

Save

Submit



## Service

Service is the third screen. This screen uses the following UI components:

- Checkboxes have a “Refresh” section. When selected, the Service sub-section appears.
  - Sub-sections are visible via a Server-side *When* rule, so when you call the refresh API, the server recalculates based upon *When* rules and returns data with the new section(s).
- Radio buttons.

**Service (S-57) NEW**

**Service** R

<b>TV/Cable Service</b>	<b>Internet Service</b>	<b>Home Phone Service</b>
<input checked="" type="checkbox"/> TV	<input checked="" type="checkbox"/> Internet	<input type="checkbox"/> Phone
TV Option	Internet Option	
<input type="radio"/> Basic	<input type="radio"/> 25 Mbps	
<input checked="" type="radio"/> Basic Plus	<input checked="" type="radio"/> 100 Mbps	
<input type="radio"/> Deluxe	<input type="radio"/> 300 Mbps	
<input type="radio"/> Premium		

Cancel Save Submit

## Notes

This is the fourth screen. This screen uses the following UI components:

- Text input
- Checkbox. If the checkbox is selected, upon submitting, the application routes the service request to a Manger workbasket. Otherwise, the application routes it to Tech.

**Service (S-57) NEW**

**Notes**

**Other Notes**

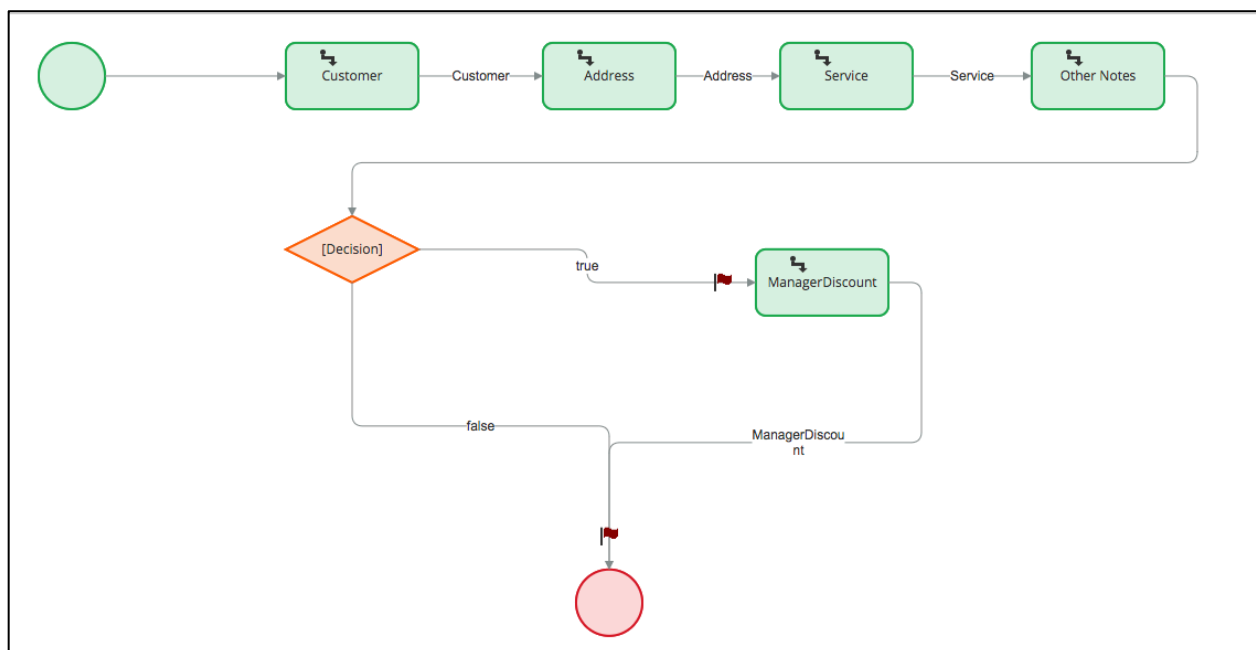
Notes

asking for \$25 off

☒ Send to Manager for Discount

Cancel Save Submit

In the flow, we have a decision based upon the checkbox as shown in the following figure:



## Manager Discount

This screen is seen by Manager. Work items appear in the Manager worklist. Manager can add a customer discount. This screen uses the following UI components:

- Read-only
- Number

**Service (S-57)** **PENDING MANAGER**

**ManagerDiscount**

Notes to Manager  
asking for \$25 off

Customer Discount  
25.00

Cancel

Save

Submit

## Service Connect

This screen is seen by Tech. Tech fulfills the request and marks the services that have been fulfilled. Tech can also update/modify the service request.

This screen uses the following UI components:

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

**Service (S-57)** **PENDING-FULFILLMENT**

### Service Connect

**Customer**  
First Name  
John  
Last Name  
Smith  
Email  
john@smith.com  
Phone number  
6178666000  
Expected Service Date  
Friday, February 15, 2019  
Discount  
\$25.00

**Chosen Services**  
TV Option  
☐ Basic  
☒ Basic Plus  
☐ Deluxe  
☐ Premium  
Internet Option  
☐ 25 Mbps  
☒ 100 Mbps  
☐ 300 Mbps  
Update Service

**Fulfillment**  
☒ TV Connected  
☒ Internet Connected  
☐ Phone Connected

Cancel

SaveSubmit

## Update Service

Clicking the **Update Service** button in the Services screen displays the Update Service screen. The Update Screen uses a local action that can update/modify the service request. You can modify the service request and confirm the changes by pressing the **Submit** button. The application routes the service request back to the Service Connect screen.

**Service (S-57) PENDING-FULFILLMENT**

**UpdateService**

**TV/Cable Service**

☒ TV

TV Option

☐ Basic

☒ Basic Plus

☐ Deluxe

☐ Premium

**Internet Service**

☒ Internet

Internet Option

☐ 25 Mbps

☒ 100 Mbps

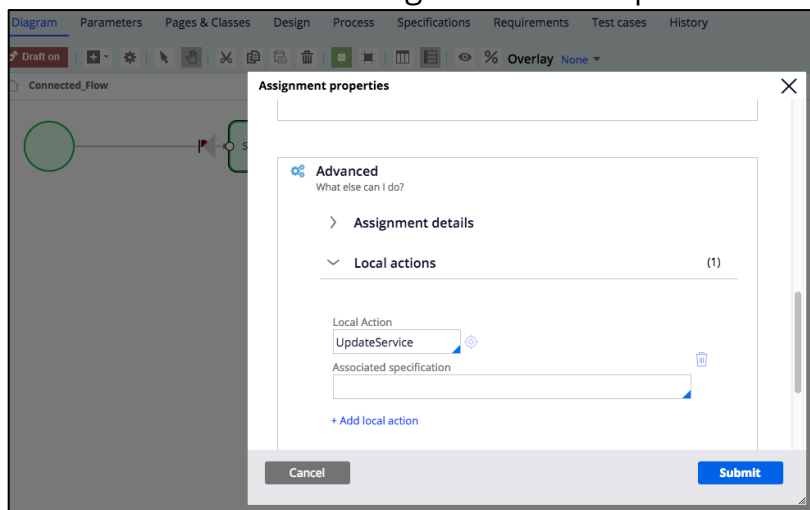
☐ 300 Mbps

**Home Phone Service**

☐ Phone

**Cancel** **Save** **Submit**

The Local action on an assignment UI component is used here.



## Confirm

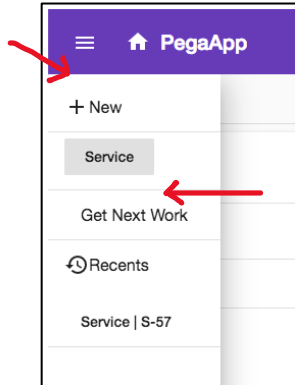
Confirm screen displays at the end when no further action is required. The Confirm harness must be overridden, as the default currently does not work with DX API.

Overridden Confirm harness with screen layout is used here.

Thank you for your submission. No further action is required.

# Step Through Angular Application

After logging into the application, select the hamburger icon, and then select **Service** to create a new Service work item.



## Customer

Customer is the first screen when logged in as “Rep”. This screen uses the following UI components:

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

A screenshot of the 'Customer' form in the PegaApp. The form is titled 'Customer' and is part of a 'New Service' case. The form is divided into two main sections: 'Customer' and 'Case details'. The 'Customer' section contains the following fields: 'First Name \*' (John), 'Middle Name' (empty), 'Last Name \*' (Smith), 'Suffix' (empty), 'Email \*' (john@smith.com), and 'Service Date' (2/15/2019). The 'Case details' section contains the following fields: 'Last updated by' (Rep.CableCo, a few seconds ago) and 'Created by' (Rep.CableCo, a few seconds ago). At the bottom of the form are three buttons: 'Cancel', 'Save', and 'Submit'.

- Autocomplete via prompt list

The screenshot shows a web application interface for a 'New Service' form. The breadcrumb navigation is 'New Service > Connected > Resolved'. The form is divided into two main sections: 'Customer' and 'Case details'. In the 'Customer' section, there are input fields for 'First Name \*' (containing 'John'), 'Middle Name', and 'Last Name \*' (containing 'Smith'). Below these is a 'Suffix' field with a dropdown menu open, showing options: 'Sr -- (Sr)', 'Jr -- (Jr)', 'III -- (III)', 'IV -- (IV)', and 'V -- (V)'. The 'Case details' section shows 'Last updated by' as 'Paul Gagnon' (2 minutes ago) and 'Created by' as 'Rep.CableCo' (28 minutes ago). At the bottom right, there are 'Save' and 'Submit' buttons.

## Error handling

- Required – local error handling in Angular App

This screenshot shows a single form field labeled 'First Name \*' in red. Below the input field, a red error message reads: 'You must enter a value'.

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

This screenshot shows a form field labeled 'Email \*' in red. The input contains the letter 'a'. Below the input field, a red error message reads: 'Not a valid email'.

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

This screenshot shows a form field labeled 'Service Date' in red. The input contains the date '2/1/2019'. To the right of the input is a calendar icon. Below the input field, a red error message reads: 'Service date MUST either be TODAY or be in the future'.



## Address

Address is the second screen. This screen uses the following UI components:

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

The screenshot shows a web application interface for an 'Address' form. At the top, there's a breadcrumb trail: 'New Service > Connected > Resolved'. The form is divided into two main sections: 'Address' and 'Case details'. The 'Address' section contains five input fields: 'Street' (with the value '1 Rogers St'), 'City' (with the value 'Cambridge'), 'State' (a dropdown menu currently showing 'MA'), 'Postal Code' (with the value '02142'), and 'Phone number' (with the value '617 866-6000' and a placeholder '(###) ###-####'). The 'Case details' section contains two fields: 'Last updated by' (with the value 'Rep.CableCo' and 'a few seconds ago') and 'Created by' (with the value 'Rep.CableCo' and 'a minute ago'). At the bottom of the form, there are three buttons: 'Cancel', 'Save', and 'Submit'.

- Drop Down via data page

This screenshot shows the same 'Address' form as the previous one, but with the 'State' dropdown menu open. The dropdown menu displays a list of states: 'KS', 'KY', 'LA', 'MA' (which is highlighted), 'MD', 'ME', and 'MI'. The rest of the form, including the 'Case details' section and the bottom buttons, remains the same as in the previous screenshot.

## Service

Service is the third screen. This screen uses the following UI components:

- Checkboxes have a “Refresh” section. When selected, the Service sub-section appears.
- Sub-sections are visible via a Server-side When rule, so when you call the refresh API, the server recalculates based upon When rules and returns data with the new section(s).
- Radio buttons.

The screenshot shows the 'Service' screen in a web application. At the top, there's a breadcrumb trail: 'New Service > Connected > Resolved'. Below this, the screen is divided into two main sections: 'Service' and 'Case details'. The 'Service' section has three sub-sections: 'TV/Cable Service', 'Internet Service', and 'Home Phone Service'. Under 'TV/Cable Service', there is an unchecked checkbox for 'TV'. Under 'Internet Service', there is an unchecked checkbox for 'Internet'. Under 'Home Phone Service', there is an unchecked checkbox for 'Phone'. The 'Case details' section shows 'Last updated by' as 'Paul Gagnon' and 'Created by' as 'Rep.CableCo'. At the bottom, there are 'Cancel', 'Save', and 'Submit' buttons.

- Selecting checkboxes causes refresh and displays the sub-sections.

This screenshot shows the 'Service' screen after the 'TV' and 'Internet' checkboxes have been selected. The 'TV' sub-section is now visible under 'TV/Cable Service', showing 'TV Option' with radio buttons for 'Basic', 'Basic Plus' (selected), 'Deluxe', and 'Premium'. The 'Internet' sub-section is visible under 'Internet Service', showing 'Internet Option' with radio buttons for '25 Mbps', '100 Mbps' (selected), and '300 Mbps'. The 'Home Phone Service' section remains unchanged with the 'Phone' checkbox unchecked. The 'Case details' section now shows 'Last updated by' as 'Rep.CableCo' and 'Created by' as 'Rep.CableCo'. The 'Cancel', 'Save', and 'Submit' buttons are still at the bottom.

## Notes

This is the fourth screen. This screen uses the following UI components:

- Text input.
- Checkbox. If the checkbox is selected, upon submit, the application routes the service request to the Manager workbasket. Otherwise, the application routes it to Tech.

Dashboard S-58 X

New Service > Connected > Resolved Actions

### Notes

### Case details

#### Other Notes

Notes

Asking for \$25 off

☒ Send to Manager for Discount

Cancel

Save Submit

#### Last updated by

Rep.CableCo  
a few seconds ago

#### Created by

Rep.CableCo  
2 minutes ago

## Manager Discount

This screen is seen by Manager. Work items appear in the Manager worklist.

Dashboard		Worklist	
		CableConnect:Managers	
Case	Status	CableConnect:Users	Urgency
S-61	Pending Manager	Service	10
Items per page: 10 1 - 1 of 1  < < > >			

Manager can add a customer discount. This screen uses the following UI components:

- Read-only
- Number (only numbers are allowed)

Dashboard

S-58 X

New Service > Connected > Resolved

Actions

ManagerDiscount

Notes to Manager  
Asking for \$25 off  
Customer Discount  
25.00  
\*Add a customer discount as XX.YY (dollars)\*

Case details

Last updated by  
Rep.CableCo  
a few seconds ago  
Created by  
Rep.CableCo  
3 minutes ago

Cancel

Save

Submit

## Service Connect

This screen is seen by Tech. Tech fulfills the request and marks the services that have been fulfilled. Tech can also update/modify the service request.

This screen uses the following UI components:

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

Dashboard

S-58 X

New Service > Connected > Resolved

Actions

Service Connect

Customer

First Name  
John

Last Name  
Smith

Email  
john@smith.com

Phone number  
617 866-6000

Expected Service Date  
Friday, February 15, 2019

Discount  
\$25.00

Chosen Services

TV Option

Basic

Basic Plus

Deluxe

Premium

Internet Option

25 Mbps

100 Mbps

300 Mbps

Update Service

Fulfillment

TV Connected

Internet Connected

Phone Connected

Case details

Last updated by

Manager.CableCo  
a few seconds ago

Created by

Rep.CableCo  
4 minutes ago

Cancel

Save

Submit

## Update Service

Clicking the **Update Service** button in the Services screen displays the Update Service screen. The Update Screen uses a local action to update/modify the service request. You can modify the service request and confirm the changes by pressing the **Submit** button. The application routes the service request back to the Service Connect screen.

Local action on an assignment is exercised here.

Dashboard

S-58 X

New Service > Connected > Resolved

Actions

Service			Case details
<b>TV/Cable Service</b>	<b>Internet Service</b>	<b>Home Phone Service</b>	
<input checked="" type="checkbox"/> TV TV Option <input type="radio"/> Basic <input checked="" type="radio"/> Basic Plus <input type="radio"/> Deluxe <input type="radio"/> Premium	<input checked="" type="checkbox"/> Internet Internet Option <input type="radio"/> 25 Mbps <input checked="" type="radio"/> 100 Mbps <input type="radio"/> 300 Mbps	<input type="checkbox"/> Phone	<b>Last updated by</b> Manager.CableCo a few seconds ago  <b>Created by</b> Rep.CableCo 4 minutes ago
<div>Cancel</div>			<div>Save</div> <div>Submit</div>

## Confirm

Confirm screen displays at the end, when no further action is required. The Confirm harness must be overridden, as the default does not work with DX API, currently.

The Overridden Confirm harness with screen layout is used here.

