



Using PCore and PConnect Public APIs '24.2

5 September 2025

CONTENTS

| | |
|---|----------|
| Accessing APIs from the PConnect object | 7 |
| Directly accessed PConnect APIs. | 7 |
| acceptSuggestion(target). | 10 |
| clearErrorMessages(errorMsgConfig). | 11 |
| createComponent(componentMeta, dataSource, index, additionalPropsToComp). | 12 |
| deRegisterAdditionalProps(deRegisterKeys). | 14 |
| getActions(). | 14 |
| getActionsApi(). | 15 |
| getCaseInfo(). | 16 |
| getCaseStages(). | 16 |
| getCaseSummary(). | 17 |
| getChildren(). | 17 |
| getComponentName(). | 18 |
| getConfigProps(destinationObject). | 19 |
| getContainerManager(). | 19 |
| getContainerName(). | 20 |
| getContextName(). | 20 |
| getCurrentClassID(). | 21 |
| getCurrentView(). | 21 |
| getDataObject(contextName). | 22 |
| getGoogleMapsAPIKey(). | 22 |
| getImagePath(imageName). | 23 |
| getInheritedProps(). | 24 |
| getListActions(). | 25 |
| getLocaleRuleName(). | 26 |
| getLocalizedValue(rawString, localePath, localeRuleKey). | 27 |
| getPageReference(). | 28 |
| getRawMetadata(). | 29 |

| | |
|---|----|
| getReferencedUser(userId). | 31 |
| getReferencedView(). | 31 |
| getReferencedViewPConnect(passOptions). | 32 |
| getServerURL(). | 34 |
| getStateProps(). | 34 |
| getTarget(). | 35 |
| getValidationApi(). | 35 |
| getValue(prop, pageReference). | 36 |
| hasChildren(). | 36 |
| ignoreSuggestion(target). | 37 |
| isBoundToState(). | 38 |
| isConditionExist(). | 39 |
| isEditable(). | 39 |
| populateAdditionalProps(configData). | 40 |
| registerAdditionalProps(additionalMeta). | 41 |
| replaceState(prop, value). | 42 |
| reportError(errorMsg, errorObj, context). | 42 |
| resolveConfigProps(configProps, destinationObject). | 43 |
| setAction(eventType, handler). | 44 |
| setInheritedProp(propName, value). | 45 |
| setPageValue(sourcePage, target, isPerform). | 46 |
| setValue(prop, value, initialValue, isImplicit, options). | 47 |
| updateState(dataObject). | 50 |
| errorMsgConfig. | 51 |
| ActionsApi class. | 52 |
| approveCase(containerItemID). | 54 |
| cancelAssignment(containerItemID). | 55 |
| cancelCreateStageAssignment(containerItemID). | 56 |
| cancelDataObject(containerItemID). | 57 |
| cancelLocalAction(containerItemID). | 58 |
| changeHandler(pConn, event). | 59 |
| createDataObject(containerItemID). | 60 |

| | |
|--|-----|
| createWork(className, options). | 61 |
| deleteCaseInCreateStage(containerItemID, ignoreCaseDeletion). | 65 |
| deleteDataObject(className, keys). | 66 |
| eventHandler(pConn, event). | 67 |
| fillFormWithAI(containerItemID). | 68 |
| finishAssignment(containerItemID, options). | 69 |
| getDataObjectActions(dataViewID, dataViewParameters). | 71 |
| getDataObjectView(className, keyProperties, options). | 72 |
| getDescendants(caseID, context). | 73 |
| getNextWork(). | 74 |
| getRecents(maxResultsToFetch). | 75 |
| ignoreDuplicateCase(containerItemID). | 76 |
| invoke(url, opts). | 77 |
| loadView(caseID, viewName, config). | 78 |
| navigateToStep(stepID, containerItemID). | 80 |
| openAssignment(assignmentID, className, options). | 81 |
| openDataObjectAction(className, keyProperties, actionID, actionName). . . | 83 |
| openEmbeddedDataModal(viewName, component, targetProperty, index, action, heading). | 84 |
| openLocalAction(actionID, options). | 86 |
| openProcessAction(actionID, options). | 90 |
| openStage(stageID). | 92 |
| openWorkByHandle(workID, className, options). | 93 |
| rejectCase(containerItemID). | 94 |
| resolveDuplicateCase(containerItemID). | 95 |
| saveAndClose(containerItemID). | 96 |
| saveAssignment(containerItemID). | 97 |
| showCasePreview(pzInsKey, configObj). | 98 |
| showData(pageName, dataContext, dataContextParameters, options). . . . | 99 |
| showDataObjectCreateView(className, viewName). | 101 |
| showDataPreview(dataContext, dataContextParameters, options). | 102 |
| showPage(pageName, className, options). | 104 |

| | |
|---|-----|
| submitDataObjectAction(containerItemID, keyProperties, actionID). | 106 |
| submitEmbeddedDataModal(containerItemID). | 107 |
| triggerFieldChange(propName, value, skipValidation). | 107 |
| updateDataObject(containerItemID, keys). | 109 |
| updateFieldValue(propName, value, options). | 110 |
| CaseInfo class. | 112 |
| getActions(). | 113 |
| getActiveFlowActionID(). | 113 |
| getAssignmentID(). | 114 |
| getAvailableProcesses(). | 114 |
| getBusinessID(). | 116 |
| getCaseStages(caseID, context). | 116 |
| getCaseTypeName(). | 117 |
| getClassName(). | 118 |
| getCurrentAssignmentViewName(). | 118 |
| getID(). | 119 |
| getKey(). | 119 |
| getName(). | 120 |
| getParentCaseInfo(). | 120 |
| isPerform(). | 121 |
| isReview(). | 121 |
| ContainerManager class. | 122 |
| activateContainerItem(containerInfo). | 123 |
| addContainerItem(containerInfo). | 123 |
| addTransientItem(containerInfo). | 124 |
| initializeContainers(containerInfo). | 125 |
| removeContainerItem(containerInfo). | 126 |
| removeTransientItem(transientItemInfo). | 127 |
| resetContainers(containerInfo). | 128 |
| updateContainerItem(containerInfo). | 128 |
| updateTransientData(transientObject). | 130 |
| ListActions class. | 130 |

| | |
|--|-----|
| clearSelectedRows(). | 131 |
| deleteAll(property, options). | 133 |
| deleteEntry(index, pageRef). | 134 |
| deletePage(property). | 136 |
| getSelectedRows(withPayload). | 136 |
| initDefaultPageInstructions(property, propertyNames, uniqueField). | 138 |
| insert(payload, index, pageRef, options). | 139 |
| reorder(fromIndex, toIndex). | 141 |
| replacePage(property, payload). | 143 |
| setSelectedRows(rows, options). | 144 |
| setVisibility(isVisible). | 146 |
| update(payload, index, options). | 147 |
| updateProperty(propertyName, value, options). | 150 |
| ValidationApi class. | 152 |
| validate(value, propertyName). | 152 |

Accessing APIs from the PConnect object

The PConnect APIs obtain or update information about each specific instance of a UI component that exists in an application at runtime.

PConnect is a context-aware object that is linked to a component. It obtains context details such as page reference, active container, Redux pointer, localization reference, related assignment, and case or data object information. Each component has its own PConnect object, and no two components share the same object. When a component is deleted or removed from the screen, its associated PConnect object is also deleted.

Use the PConnect object when there is a known context that provides necessary information about the component being rendered from the Constellation JavaScript Engine.

PConnect APIs can also be used to create new instances of UI components and determine how a component's rendering should be updated.

- **Directly accessed PConnect APIs**
- **APIs in the ActionsApi class**
- **APIs in the CaseInfo class**
- **APIs in the ContainerManager class**
- **APIs in the ListActions class**
- **APIs in the ValidationApi class**

Directly accessed PConnect APIs

Access APIs directly from the PConnect object.



- `acceptSuggestion(target)`
- `clearErrorMessages(errorMsgConfig)`
- `createComponent(componentMeta, dataSource, index, additionalPropsToComp)`
- `deRegisterAdditionalProps(deRegisterKeys)`
- `getActions()`
- `getActionsApi()`
- `getCaseInfo()`
- `getCaseStages()`
- `getCaseSummary()`
- `getChildren()`
- `getComponentName()`
- `getConfigProps(destinationObject)`
- `getContainerManager()`
- `getContainerName()`
- `getContextName()`
- `getCurrentClassID()`
- `getCurrentView()`
- `getDataObject(contextName)`
- `getGoogleMapsAPIKey()`
- `getImagePath(imageName)`

- `getInheritedProps()`
- `getListActions()`
- `getLocaleRuleName()`
- `getLocalizedValue(rawString, localePath, localeRuleKey)`
- `getPageReference()`
- `getRawMetadata()`
- `getReferencedUser(userId)`
- `getReferencedView()`
- `getReferencedViewPConnect(passOptions)`
- `getServerURL()`
- `getStateProps()`
- `getTarget()`
- `getValidationApi()`
- `getValue(prop, pageReference)`
- `hasChildren()`
- `ignoreSuggestion(target)`
- `isBoundToState()`
- `isConditionExist()`
- `isEditable()`
- `populateAdditionalProps(configData)`

- **registerAdditionalProps(additionalMeta)**
- **replaceState(prop, value)**
- **reportError(errorMsg, errorObj, context)**
- **resolveConfigProps(configProps, destinationObject)**
- **setAction(eventType, handler)**
- **setInheritedProp(propName, value)**
- **setPageValue(sourcePage, target, isPerform)**
- **setValue(prop, value, initialValue, isImplicit, options)**
- **updateState(dataObject)**
- **errorMsgConfig**

acceptSuggestion(target)


Accepts the value stored in the Suggestions Context for a field.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|---------------|--------|--|--------------------------|
| target | string | The field whose value stored in the Suggestions Context is accepted. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <div>  NOTE: If the target is not provided, it is determined automatically. </div> | |

Usage example

In this example, the API accepts the value stored in the Suggestions Context for the `Property1` field.

```
getPConnect().acceptSuggestion('.Property1');
```

For more information on the Suggestions Context, see the [getSuggestionsContext\(context\)](#) API.

clearErrorMessages(errorMsgConfig)


Deletes the error messages associated with a specific context or app container.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------------|----------------|---|--------------------------|
| errorMsgConfig | errorMsgConfig | The object containing information specifying the error messages that are deleted. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|---|----------|
| | |  NOTE: For more information, see errorMsgConfig . | |

Usage example

In the following example, the API deletes error messages at the `app/primary` level that belong to the `HTTP` category.

```
getPConnect().clearErrorMessages({category: "HTTP", context: "app/primary"});
```

In the following example, the API deletes error messages at the app level that belong to the `HTTP` category.

```
getPConnect().clearErrorMessages({category: "HTTP"});
```

In the following example, the API deletes error messages related to the `firstName` property.

```
getPConnect().clearErrorMessages({property: ".firstName"});
```

createComponent(componentMeta, dataSource, index, additionalPropsToComp)

Creates a component whose type is provided as part of the meta object and uses the other parameters to resolve the properties and get the context.



NOTE: Use this API when a composite component needs to create other components dynamically.

Returns

A newly created instance of a component.

Parameters

| Name | Type | Description | Required |
|------------------------------|--------|--|--------------------------|
| componentMeta | object | The metadata of the component to be created. This mainly contains the following properties: <ul style="list-style-type: none"> <code>type</code> - The name of the component <code>config</code> - The configuration props as required by the component to be created. <code>options</code> - The optional props for the component to be created. | <input type="checkbox"/> |
| dataSource | string | The source reference to the Redux store. This contains properties that will be input to the created component. | <input type="checkbox"/> |
| index | string | If the dataSource is a list, this is the index position that will be input to the created component. | <input type="checkbox"/> |
| additionalPropsToComp | object | The additional properties that are passed as input to the created component apart from the configured values. | <input type="checkbox"/> |

Usage example

In this example, a button is created with a label that is derived from the first index position of the `D_Employees.pxResults` list.

```
getPConnect().createComponent({
  type: "Button",
  config: {
    label: "@P.FirstName"
  }
},
"D_Employees.pxResults",
1, {
  active: true
}
);
```

deRegisterAdditionalProps(deRegisterKeys)

Deregisters the input properties passed from metaConfig. When you call the [getConfigProps\(destinationObject\)](#) API, only the registered properties are obtained.

Returns

The Boolean value `true` after deregistering the input properties.

Parameters

| Name | Type | Description | Required |
|-----------------------|---------------------------|---|--------------------------|
| deRegisterKeys | array.<(string number)> | The property that needs to be deregistered. | <input type="checkbox"/> |

Usage example

In this example, the API deregisters the properties in `label`.

```
const props = getPConnect().deRegisterAdditionalProps(['label']);
```

getActions()

Obtains the actions object that is used to configure a component's event actions.

Returns

An actions object.

Parameters

This API does not have parameters.

Usage example

In this example, the API provides the currency component's action-set.

```
const actions = getPConnect().getActions();  
return <Currency value={value} label={label} {...actions} />;
```

getActionsApi()

Obtains an entry point to the ActionsApi object that contains the APIs to handle actions in the Constellation architecture.

To view the APIs in the ActionsApi class, see [APIs in the ActionsApi class](#).

Returns

The ActionsApi object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ActionsApi object.



```
const actionsAPI = pConn.getActionsApi();
```

getCaseInfo()

Obtains an entry point to the CaseInfo object that contains the APIs to handle the information related to a case.

To view the APIs in the CaseInfo class, see [APIs in the CaseInfo class](#).

Returns

The CaseInfo object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the CaseInfo object.

```
const caseInfo = getPConnect().getCaseInfo();
```

getCaseStages()

Obtains the stages within a case.

Returns

An object containing the stages in a case in the following structure:

```
{ ID: "PRIM1" entryTime: "20200520T093921.867 GMT" links: {...} name: "Collect Info"
  type: "Primary" visited_status: "active" }.
```


Parameters

This API does not have parameters.

Usage example

In this example, the API returns the stages that are associated with a case.

```
const stages = this.getCaseStages();
```

getCaseSummary()

Returns the details of the case from the metadata of the loaded application.

Returns

An object containing the details of the case in the following structure:

```
{ routedTo:"user@constellation.com" urgency:"10" caseID:"OPB1HW-SPACETRA-WORK RA-219" name:"Request approval" caseClass:"OPB1HW-SpaceTra-Work-RequestApproval" ID:"ASSIGN-WORKLIST OPB1HW-SPACETRA-WORK RA-219!REQUEST_FLOW_0" }.
```

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the case summary using the case ID from the metadata of the loaded application.

```
const caseInfo = this.getCaseSummary();
```

getChildren()

Obtains the list of children for the current component. The iterator is also reset to the beginning of the index.

Returns

An object containing the list of children.

Parameters

This API does not have parameters.

Usage example

In this example, the API retrieves the list of child components from the current component. This list is stored in the children variable.

```
const children = getPConnect().getChildren();
```

getComponentName()

Obtains the name of the current component.

Returns

The name of the current component as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the current component.

```
const component = getPConnect().getComponentName();
```

getConfigProps(destinationObject)

Obtains the resolved configuration of a component.

Returns

An object containing the resolved configuration of a component.

Parameters

| Name | Type | Description | Required |
|--------------------------|--------|--|----------|
| destinationObject | object | The object to which the resolved configuration properties are added. | ☐ |

Usage example

In this example, the API returns the resolved configuration of a component.

```
const props = getPConnect().getConfigProps();
```

getContainerManager()

Provides an entry point to the ContainerManager object that contains the APIs to manage the containers used in complex components.

To view the APIs in the ContainerManager class, see [APIs in the ContainerManager class](#).

Returns

The ContainerManager object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ContainerManager object.

```
const containerManager = getPConnect().getContainerManager();
```

getContainerName()

Obtains the name of the container associated with the current component.

Returns

The name of the container as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns `primary` as the name of the container.

```
const { containerName = "primary" } = getPConnect().getContainerName();
```

getContextName()

Obtains the name of the context under which the current component was rendered.

Returns

The name of the context as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the context under which the current component was rendered.

```
const context = getPConnect().getContextName();
```

getCurrentClassID()

Obtains the class ID corresponding to the current field.

Returns

The class ID as a string.

Parameters

This API has no parameters.

Usage example

In this example, the API returns the class ID corresponding to the current field.

```
const firstNameClassId = getPConnect().getCurrentClassID();
```

getCurrentView()

Obtains the name of the view under which the current component was rendered.

Returns

The name of the view as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the view under which the current component was rendered.

```
const viewName = getPConnect().getCurrentView();
```

getDataObject(contextName)

Obtains the data associated with the current context from the Store.

Returns

The data as an object.

Parameters

| Name | Type | Description | Required |
|--------------------|--------|--|--------------------------|
| contextName | string | The name of the current context whose associated data is obtained. | <input type="checkbox"/> |

Usage example

In this example, the API obtains the data associated with the current context from the Store and stores it in the data variable.

```
const data = pConn.getDataObject();  
const { caseInfo, pulse } = data;
```

getGoogleMapsAPIKey()

Used in the Location component of the DX Components layer to obtain the API key for the Google Maps application from the DSS setting (uiengine/map/googleapikey).

Returns

The API key for the Google Maps application as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the API key to render Google Maps.

```
let apiKey = getPConnect().getGoogleMapsAPIKey();
```

getImagePath(imageName)

Obtains the resolved path of an image.

Returns

The API key for the image path as a string.

Parameters

| Name | Type | Description | Required |
|-----------|--------|---|----------|
| imageName | string | The name of the image whose resolved path must be obtained. | ☐ |

Usage example

In this example, the API obtains the resolved path to the `launcher.png` image, which is then stored in `imagePath`.

```
const imagePath = getPConnect().getImagePath("launcher.png");
```

getInheritedProps()

Obtains the properties that are set in the parent metadata node within the `inheritedProps` metadata structure. The `inheritedProps` metadata allows cascading of values in the view metadata, which results in better reuse of the properties.

Returns

The inherited properties as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the value of the `label` property inherited from the parent metadata and overrides the value of the `label` property set in the child view.

```
// Parent reference metadata
{
  "type": "reference",
  "config": {
    "name": "Address",
    "inheritedProps": [
      {
        "prop": "label",
```



```

    "value": "@L Shipping address"
  }
]
}
}

```

// Child reusable Address view

```

{
  "name": "Address",
  "type": "View",
  "config": {
    "template": "OneColumn",
    "ruleClass": "Data-Address"
    "label": "Generic address"
  },
  "children": [
    {
      "name": "A",
      "type": "Region",
      "children": []
    }
  ]
}

```

// During rendering the Address child view, you can retrieve the inherited label as follows:

```
const propsToUse = { label, ...getPConnect().getInheritedProps() };
```

// This results in propsToUse.label being assigned the “Shipping address” value. This inherits the value that was set on the parent (if the parent had a “label” property) and overrides any “label” property that was set on the child view.

getListActions()

Obtains an entry point to the ListActions class that contains APIs that handle list related actions for components that display page list data and allow modifications on the list.

To view the APIs in the ListActions class, see [APIs in the ListActions class](#).

Returns

The ListActions object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ListActions object.

```
getPConnect().getListActions();
```

getLocaleRuleName()

Obtains the locale reference rule name referenced by the environment object for the field values.

Returns

The locale reference rule name as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns `VEHICLE - !VIEW!VEHICLEDETAILS` if the current environment object corresponds to a view named `VehicleDetails` in the `Work` class. The `VEHICLE - !VIEW!VEHICLEDETAILS` localization rule is mapped 1-1 to the `VehicleDetails` view.

```
const localeRuleName = getPConnect().getLocaleRuleName();
```


getLocalizedValue(rawString, localePath, localeRuleKey)

Obtains the localized value of a string.

Returns

The localized value as a string.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|--------------------------|
| rawString | string | The raw string whose localized value must be obtained. | <input type="checkbox"/> |
| localePath | string | The lookup path of the raw string. <div> NOTE: The default value is <code>fields</code>.</div> | <input type="checkbox"/> |
| localeRuleKey | string | The rule key in the localization store. | <input type="checkbox"/> |

Usage example

In this example, the API returns `Primeiro nome` for the `WORK-!VIEW!CANDIDATEINFO` rule.

```
const localizedString = getPConnect().getLocalizedValue("First Name", "fields", "WORK-!VIEW!CANDIDATEINFO");

//The localization store has the following structure
{
  "WORK-!VIEW!CANDIDATEINFO.json" : {
    "fields" : {
      "First Name" : "Primeiro nome",
      "Last Name" : "Último nome"
    }
  }
}
```

getPageReference()

Obtains the data reference path of the store that contains the data value of the current component.

Returns

The data reference path as a string.

Parameters

This API does not have parameters.

Usage example

- When a portal is loaded at the app level, the value of `pageReference` is `pyPortal`.
- When a single component (such as FirstName, LastName, etc) is loaded, the value of `pageReference` is `caseInfo.content`.
- When embedded components under embedded view (such as ZipCode, StreetName, etc., that are under the Address view) are loaded, the value of `pageReference` is `caseInfo.content.Address`.

getRawMetadata()

Obtains the complete metadata of the component in its raw and unresolved format.

NOTE: Use this API only in advanced use cases where a component needs the complete unresolved metadata. In most cases, use the `getConfigProps(destinationObject)` API to get the resolved configuration metadata of the component. The format and content of the raw metadata returned may change as the Constellation DX API changes.

Returns

The raw metadata of the component as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the raw unresolved metadata for a simple `Table` component.

```

{
  "type": "Table",
  "config": {
    "referenceList": "D_pyMyWorkList.pxResults",
    "rowClickAction": "openAssignment",
    "personalizationId": "pyMyWork-D_pyMyWorkList",
    "fields": [
      {
        "type": "Text",
        "config": { "label": "@L Priority", "formatType": "number", "value": "@P.pxUrgencyAssign" }
      },
      {
        "type": "Text",
        "config": { "label": "@L ID", "value": "@P.pxRefObjectInsName" }
      },
      {
        "type": "Text",
        "config": { "label": "@L Label", "value": "@P.pyLabel" }
      },
      {
        "type": "DateTime",
        "config": { "sortDirection": "DESC", "sortPriority": "1", "label": "@L Creation date/time", "value": "@P.pxCreateDateTime" }
      },
      {
        "type": "Text",
        "config": { "label": "@L Status", "value": "@P.pyAssignmentStatus" }
      }
    ]
  }
}

```

getReferencedUser(userId)

Obtains the information associated with a referenced user.

Returns

The information as an object.

Parameters

| Name | Type | Description | Required |
|--------|--------|---|--------------------------|
| userId | string | <p>The unique identifier of a user.</p> <p>Example:</p> <pre>"abc": { "UserName": "first_name1" }</pre> | <input type="checkbox"/> |

Usage example

In this example, the API returns the information associated with the user whose user ID is `abc`.

```
getPConnect().getReferencedUser(abc);
```

getReferencedView()

Obtains the view metadata of a referenced view.

Returns

The referenced view as an object.



Parameters

This API does not have parameters.

Usage example

In this example, the API returns `viewMetadata` as the metadata object of the `Address` view.

```
const meta = {  
  meta: {  
    type: 'reference',  
    config: {  
      type: 'view',  
      name: 'Address'  
    }  
  }  
}  
  
const c11nEnv = createC11nEnv(meta).getPConnect();  
const viewMetadata = c11nEnv.getReferencedView();
```

getReferencedViewPConnect(passOptions)

Obtains the PConnect object associated with the referenced view.

Returns

The object associated with the referenced view.

Parameters

| Name | Type | Description | Required |
|--------------------|---------|--|--------------------------|
| passOptions | boolean | <p>The flag that determines if the reference options will be passed to the view.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • Set passOptions to <code>true</code> if you want to pass the reference options to the view. • Set passOptions to <code>false</code> if you do not want to pass the reference options to the view. </div> | <input type="checkbox"/> |

Usage example

In this example, the API returns the PConnect object associated with the `Address` view.

```
const meta = {
  meta: {
    type: 'reference',
    config: {
      type: 'view',
      name: 'Address'
    }
  }
}
```

```
const pConnect = createC11nEnv(meta);  
const pConn = pConnect.getPConnect().getReferencedViewPConnect();
```

getServerURL()

Obtains the current server's URL.

Returns

The server URL as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the current server's URL.

```
let serverurl = getPConnect().getServerURL();
```

getStateProps()

Obtains the properties that are bound to the state of a Store. These properties represent the current state of the Store.

Returns

The properties as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the properties bound to the state of a Store, which are then stored in props.

```
const props = getPConnect().getStateProps();
```

getTarget()

Obtains the fully qualified name of the target container with which a component is associated.

Returns

The name of the target container as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns `app/primary` as the name of the target container.

```
const target = getPConnect().getTarget();
```

getValidationApi()

Obtains an entry point to the ValidationApi object that contains APIs that handle validations on a field.

To view the APIs in the ValidationApi class, see [APIs in the ValidationApi class](#).

Returns

The ValidationApi object.



Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ValidationApi object.

```
const validationApi = pConn.getValidationApi();
```

getValue(prop, pageReference)

Obtains the value of the property from the Redux Store.

Returns

The value of the property as a string.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|---|--------------------------|
| prop | string | The name of the property whose value is obtained. | <input type="checkbox"/> |
| pageReference | string | The path of the property. | <input type="checkbox"/> |

Usage example

In this example, the API obtains the value of the `firstName` property from the Redux Store.

```
const firstName = getPConnect().getValue('.firstName');
```

hasChildren()

Determines if the current component has children.

Returns

The Boolean value `true` if the current component has children.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value, which denotes if the current component has children.

```
const componentHasChildren = getPConnect().hasChildren();
```

ignoreSuggestion(target)


Ignores the value stored in the Suggestions Context for a field.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|---------------|--------|---|--------------------------|
| target | string | The field whose value stored in the Suggestions Context is ignored. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <div>  NOTE: If the target is not provided, it is determined automatically. </div> | |

Usage example

In this example, the API ignores the value stored in the Suggestions Context for the `Property1` field.

```
getPConnect().ignoreSuggestion('.Property1');
```

For more information on the Suggestions Context, see the [getSuggestionsContext\(context\)](#) API.

isBoundToState()

Determines if the current component is bound to a Redux State.

Returns

The Boolean value `true` if the current component is bound to a Redux State.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value, which denotes if the current component is bound to a Redux State.

```
const isStateBound = getPConnect().isBoundToState();
```

isConditionExist()

Determines if the current component is associated with a condition (such as an expression or when condition).

Returns

The Boolean value `true` if the current component is associated with a condition.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value, which denotes if the current component is associated with a condition.

```
const isConditionConfigured = getPConnect().isConditionExist();
```

isEditable()

Determines if a component can be edited.

Returns

The Boolean value `true` if the component can be edited.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value, which denotes if the component can be edited.

```
const isEditable = pConn.isEditable();
```

populateAdditionalProps(configData)

Populates the specified object with additional properties that are specific to the current component but are not part of the component's configuration.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-------------------|--------|---|--------------------------|
| configData | object | The object that needs to be populated with additional properties. | <input type="checkbox"/> |

Usage example

In this example, after the **populateAdditionalProps(configProps)** API is called, the value of the `configProps` object is updated based on the type of the component.

```
const configProps = { label: 'label-name' }  
getPConnect().populateAdditionalProps(configProps);
```

The possible keys added to the `configProps` object are:

- `configObject.pageMessages (string)` - This key contains page messages that are added for the `case_view` component.
- `configObject.validateMessage (string)` - This key contains validation messages for components that are editable.
- `configObject.infoMessage (string)` - This key contains warning messages for components that are editable.
- `configObject.status (string)` - This key contains the string values that indicate the type of the message. The values can be either error or warning.
- `configObject.httpMessages (string)` - This key contains application level HTTP messages.
- `configObject.caseMessages (string)` - This key contains cases level messages.
- `configObject.stages (object)` - This key contains information about stages.
- `configObject.fieldMetadata (object)` - This key contains the information about the additional configuration that is configured in PRPC.

registerAdditionalProps(additionalMeta)

Resolves the input properties and registers them to the configuration object used for component rendering. When you call the [getConfigProps\(destinationObject\)](#) API, the resolved properties are obtained.

Returns

The resolved properties as an object.

Parameters

| Name | Type | Description | Required |
|-----------------------|--------|--|----------|
| additionalMeta | object | The properties that need to be resolved. | ☐ |

Usage example

In this example, the API resolves the `FirstName` property, obtains the first name, and registers it to label.

```
const props = getPConnect().registerAdditionalProps({ label: `@P.FirstName` });
```

replaceState(prop, value)

Replaces the state of a property or page in the Redux Store.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|--------------|--------|--|--------------------------|
| prop | string | The name of the property or page whose value needs to be replaced. | <input type="checkbox"/> |
| value | object | The value that must be assigned to the property or page. | <input type="checkbox"/> |

Usage example

In this example, the API replaces the `address` page.

```
getPConnect().replaceState('.address', {
  apartment: "Nivee Heights"
  streetName: "Park Street",
});
```

reportError(errorMsg, errorObj, context)

Saves the error information in the Redux Store. This is applicable to all errors that need to be handled at the infrastructure level.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| errorMsg | string | The error message to be logged to Redux Store. | <input type="checkbox"/> |
| errorObj | object | The error object that contains information about the error. | <input type="checkbox"/> |
| context | string | The path in the Redux store where you want to save the error information. | <input type="checkbox"/> |

Usage example

In this example, the error message is stored in the Redux Store.

```
getPConnect().reportError(  
  ': Error occurred during ajax call at fetchMessages API :',  
  {  
    errorClassification: 'Invalid inputs'  
    errorDetails: {  
      localizedValue: 'localized error message'  
    }  
  }  
);
```

resolveConfigProps(configProps, destinationObject)

Resolves the input properties and obtains the resolved properties.

Returns

The resolved properties as an object.

Parameters

| Name | Type | Description | Required |
|--------------------------|--------|---|----------|
| configProps | object | The properties that need to be resolved. | ☐ |
| destinationObject | object | The object to which the input properties must be added. | ☐ |

Usage example

In this example, the API resolves the properties in `@P.FirstName` and obtains the first name.

```
const props = getPConnect().resolveConfigProps({ label: `@P.FirstName` });
```

setAction(eventType, handler)

Assigns a new action (based on the event type and handler) to the list of the actions associated with the current component.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|------------------|----------|--|--------------------------|
| eventType | string | The type of the event that determines the new action to be added to the list of actions. | <input type="checkbox"/> |
| handler | function | The function that must be associated with the event type. | <input type="checkbox"/> |

Usage example

In this example, the API associates the change handler to the `onChange` event type and assigns the new action to the list of the actions associated with the current component.

```
setAction("onChange", changeHandler);
```

setInheritedProp(propName, value)

Assigns a value to the inherited property and propagates the property to its children.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| propName | object | The name of the inherited property to which a value has to be assigned. | <input type="checkbox"/> |
| value | any | The value to be assigned to the inherited property. | <input type="checkbox"/> |

Usage example

In this example, the API assigns the `DETAILS` value to the `displayMode` property.

```
getPConnect().setInheritedProp("displayMode", "DETAILS");
```

setPageValue(sourcePage, target, isPerform)

Assigns the values in a source page object to a target property.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-------------------|---------|--|--------------------------|
| sourcePage | object | The object containing values that must be assigned to a target property. | <input type="checkbox"/> |
| target | string | The name of the property that is assigned values from the source page object. | <input type="checkbox"/> |
| isPerform | boolean | <p>The flag that determines if page instructions can be generated for form submission.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> The default value is <code>false</code>. </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <ul style="list-style-type: none"> • If isPerform is set to <code>true</code>, page instructions are generated for form submission if the source page object contains a list property. • If isPerform is set to <code>false</code>, page instructions are not generated for form submission. | |

Usage example

In this example, the API assigns the values in the `sourcePage` object to the `EmbeddedPage` property.

```
const sourcePage = {street: 'ABC', city: 'DEF'};
getPConnect().setPageValue(sourcePage, '.EmbeddedPage', true);
```

setValue(prop, value, initialValue, isImplicit, options)

Assigns a value to a property in the Redux Store.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|---------------------|---------|--|--|
| prop | string | The name of the property to which a value must be assigned. | <input type="checkbox"/> |
| value | string | The value that must be assigned to a property. | <input type="checkbox"/> |
| initialValue | string | The default value assigned to the property. | <input type="checkbox"/> <div> NOTE: This parameter is applicable only when the property is a dropdown control that has a default value. </div> |
| isImplicit | boolean | The flag that determines if a recorded change that has been made to the property should be deleted before assigning a value to the property. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div><div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div><div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div> <div><div><div><div><div><div></div><div></div></div><div><div><div></div><div></div></div></div><div><div><div><</div></div></div></div></div></div></div> | |

| Name | Type | Description | Required |
|-------------------------|---------|--|--------------------------|
| isArrayDeepMerge | boolean | <p>The flag that decides if the values within an array must be deep merged or replaced in the Store.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • Set isArrayDeepMerge to <code>true</code> to deep merge the values within an array in the Store. • Set isArrayDeepMerge to <code>false</code> to replace the values within an array in the Store. </div> | <input type="checkbox"/> |

Usage example

In this example, the API assigns the `lorem ipsum` value to the `firstName` property.

```
getPConnect().setValue('.firstName', 'lorem ipsum');
```

updateState(dataObject)

Performs a deep merge of a data object to the Redux State.



NOTE: If an element with the same key is present in both the data object and the Redux State, the value of the data object will be updated in the Redux State.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-------------------|--------|--|--------------------------|
| dataObject | object | The object that must be merged to the Redux State. | <input type="checkbox"/> |

Usage example


In this example, the API merges the value of the `firstName` key to the Redux State.

```
getPConnect().updateState({
  firstName: "Test"
});
```

errorMsgConfig

Use this JSON configuration object to specify error messages in the store. It is used as a parameter in the [clearErrorMessages\(errorMsgConfig\)](#) API to indicate the messages that should be deleted.

Parameters

| Name | Type | Description | Required |
|---|--------|---|--------------------------|
| category | string | The type of error message that needs to be deleted. | <input type="checkbox"/> |
| <div>  NOTE: </div> | | | |

| Name | Type | Description | Required |
|-----------------|--------|--|--------------------------|
| | | <ul style="list-style-type: none"> • Use the HTTP category for app or top-level messages. • Use the PAGE category for page-level messages. | |
| context | string | <p>The context that contains the error messages to be deleted.</p> <p>NOTE: If the context is not provided, the error messages associated with the app container are deleted.</p> | <input type="checkbox"/> |
| property | string | The name of the property related to the error message. | <input type="checkbox"/> |

APIs in the ActionsApi class

Use the APIs in the ActionsApi class to handle actions in the Constellation architecture.

- **approveCase(containerItemID)**
- **cancelAssignment(containerItemID)**
- **cancelCreateStageAssignment(containerItemID)**
- **cancelDataObject(containerItemID)**
- **cancelLocalAction(containerItemID)**

- **changeHandler(pConn, event)**
- **createDataObject(containerItemID)**
- **createWork(className, options)**
- **deleteCaseInCreateStage(containerItemID, ignoreCaseDeletion)**
- **deleteDataObject(className, keys)**
- **eventHandler(pConn, event)**
- **fillFormWithAI(containerItemID)**
- **finishAssignment(containerItemID, options)**
- **getDataObjectActions(dataViewID, dataViewParameters)**
- **getDataObjectView(className, keyProperties, options)**
- **getDescendants(caseID, context)**
- **getNextWork()**
- **getRecents(maxResultsToFetch)**
- **ignoreDuplicateCase(containerItemID)**
- **invoke(url, opts)**
- **loadView(caseId, viewName, config)**
- **navigateToStep(stepID, containerItemID)**
- **openAssignment(assignmentID, className, options)**
- **openDataObjectAction(className, keyProperties, actionID, actionName)**
- **openEmbeddedDataModal(viewName, component, targetProperty, index, action, heading)**

- **openLocalAction(actionID, options)**
- **openProcessAction(actionID, options)**
- **openStage(stageID)**
- **openWorkByHandle(workID, className, options)**
- **rejectCase(containerItemID)**
- **resolveDuplicateCase(containerItemID)**
- **saveAndClose(containerItemID)**
- **saveAssignment(containerItemID)**
- **showCasePreview(pzInsKey, configObj)**
- **showData(pageName, dataContext, dataContextParameters, options)**
- **showDataObjectCreateView(className, viewName)**
- **showDataPreview(dataContext, dataContextParameters, options)**
- **showPage(pageName, className, options)**
- **submitDataObjectAction(containerItemID, keyProperties, actionID)**
- **submitEmbeddedDataModal(containerItemID)**
- **triggerFieldChange(propName, value, skipValidation)**
- **updateDataObject(containerItemID, keys)**
- **updateFieldValue(propName, value, options)**

approveCase(containerItemID)

Performs the approval action for the approval step configured in a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|----------|
| containerItemID | object | The ID of the container item that contains information about the current case. | ☐ |

Usage example

In this example, the success callback is called if the approval action is successful.

```
const approveCasePromise = getPConnect().getActionsApi().approveCase("app/primary_1/workarea_1");
approveCasePromise.then(() => {
  // approve case success handling
}).catch(() => {
  // approve case failure handling
});
```

cancelAssignment(containerItemID)

Cancels the current assignment.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|----------|
| containerItemID | string | The ID of the container item that contains information about the current assignment. | ☐ |

Usage example

In this example, the success callback is called if the current assignment is canceled.

```
const cancelAssignmentPromise = getPConnect().getActionsApi().cancelAssignment("
app/primary_1/workarea_1");
cancelAssignmentPromise.then(() => {
  // cancel assignment success handling
}).catch(() => {
  // cancel assignment failure handling
});
```

cancelCreateStageAssignment(containerItemID)

Cancels the current assignment in the create stage.

NOTE:



- If the current assignment is the first assignment in the create stage, the form is not dirty, and an action is triggered from the modal dialog, the case is deleted by calling the [deleteCaseInCreateStage\(containerItemID, ignoreCaseDeletion\)](#) API.
- If the current assignment is in the create stage, the form is dirty, and an action is triggered from the modal dialog, an alert dialog containing the Save and close, Continue working, and Delete buttons appears.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|----------|
| containerItemID | string | The ID of the container item that contains information about the current assignment in the create stage. | ☐ |

Usage example

In this example, the success callback is called if the current assignment is canceled in the create stage.

```
const cancelCreateStageAssignmentPromise = getPConnect().getActionsApi().cancelCreateStageAssignment("app/modal_3");
cancelCreateStageAssignmentPromise.then(() => {
  // cancel create stage assignment success handling
}).catch(() => {
  // cancel create stage assignment failure handling
});
```

cancelDataObject(containerItemID)

Cancels the creation of a data object whose create view is loaded in the specified container item.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|--------------------------|
| containerItemID | string | The ID of the container item containing the create view of the data object. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the creation of the data object is canceled.

```
const cancelCreateDataObjectPromise = getPConnect().getActionsApi().cancelDataObject("app/modal_3");
cancelCreateDataObjectPromise.then(() => {
  // cancel creation of data object success handling
}).catch(() => {
  // cancel creation of data object failure handling
});
```

cancelLocalAction(containerItemID)

Cancels the local action that is associated with the container item ID of the current component.

NOTE:



- If the local action is launched, the form is not dirty, and an action is triggered from the modal dialog, then the modal dialog is closed.
- If the local action is not launched, the form is dirty, and an action is triggered from the modal dialog, an alert dialog containing a message and the Go back and Discard buttons appears.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|----------|
| containerItemID | string | The ID of the container item associated with the current component. | ☐ |

Usage example

In this example, the success callback is called if the local action is canceled.

```
const cancelLocalAction = getPConnect().getActionsApi().cancelLocalAction("app/modal_3");
cancelLocalAction.then(() => {
  // cancel local action success handling
}).catch(() => {
  // cancel local action failure handling
});
```

changeHandler(pConn, event)

Runs a change handler for the given context of the component through the Action Processor.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|--------------|--------|--|--------------------------|
| pConn | object | The PConnect object associated with the handler being run. | <input type="checkbox"/> |
| event | object | The Document Object Model (DOM) event object associated with the change. | <input type="checkbox"/> |

Usage example

In this example, the API handles the change event triggered by the UI component.

```
getPConnect().getActionsApi().changeHandler(pConnObject, event);
```

createDataObject(containerItemID)

Creates a data object whose create view is loaded in the specified container item.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|--------------------------|
| containerItemID | string | The ID of the container item containing the create view of the data object. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the data object is created.

```
const createDataObjectPromise = getPConnect().getActionsApi().createDataObject("app/modal_3");
createDataObjectPromise.then(() => {
  // create data object success handling
}).catch(() => {
  // create data object failure handling
});
```

createWork(className, options)

Creates a work object for a specified class and flow type and displays the work object in a target container.

Supported by Pega Mobile Client™. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.

Returns


A Promise associated with the action.

Parameters


| Name | Type | Description | Required |
|------------------|--------|---|--------------------------|
| className | string | Name of the case class for which the work object should be created. | <input type="checkbox"/> |
| options | object | The JavaScript object that contains the properties required for creating the work object. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|--------------------------------|---------|--|--------------------------|
| flowType | string | <p>The type of flow based on which the work object is created.</p> <div> <i>i</i> <p>NOTE: The default value of this property is <code>pyStartCase</code>.</p> </div> | <input type="checkbox"/> |
| containerName | string | <p>The name of the container that displays the created work object.</p> <p>Example:</p> <pre>{flowType: "pyStartCase", containerName: "primary"}</pre> | <input type="checkbox"/> |
| openCaseViewAfterCreate | boolean | <p>The flag that determines if the new case view should be displayed when a case is created.</p> <div> <i>i</i> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>true</code>. • Set openCaseViewAfterCreate to <code>true</code> if the new case view should be displayed when a case is created. • Set openCaseViewAfterCreate to <code>false</code> if the new case </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|-----------------------|--------|---|--------------------------|
| | | view should not be displayed when a case is created. | |
| modalOptions | object | The JavaScript object that contains the properties required to customize the modal display. | <input type="checkbox"/> |
| remoteCaseMeta | object | The JSON object containing starting fields to be set for creating a traditional remote case. | <input type="checkbox"/> |
| startingFields | object | <p>The JSON object that contains the fields to be set while creating a case.</p> <p>NOTE: In order for startingFields to be passed in this API, include the fields in the <code>AllowedStartingFields</code> data transform.</p> <p> For more information, see Adding fields while creating cases in Constellation DX API.</p> | <input type="checkbox"/> |

The following table contains the properties of the **modalOptions** object:

| Name | Type | Description | Required |
|-----------------|---------|--|--------------------------|
| dockable | boolean | <p>The flag that determines where the modal is positioned in the screen.</p> <div>  <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • Set dockable to <code>true</code> if the modal should be displayed at the bottom right corner of the screen. • Set dockable to <code>false</code> if the modal should be displayed at the center of the screen. </div> | <input type="checkbox"/> |

Usage example

In this example, the API creates a work object within the `primary` container with the `OPB1HW-SpaceTra-Work-RequestApproval` class and the `pyStartCase` flow type.

```
const options = {
  flowType: "pyStartCase",
  containerName: "primary",
  startingFields: {
    FirstName: "Adam",
    LastName: "Smith",
    Vehicle: {
      Make: "Honda",
```



```

        Model: "Accord"
    }
}
};

const createWorkPromise = getPConnect().getActionsApi().createWork("OPB1HW-SpaceTra-Work-RequestApproval", options);
createWorkPromise.then(() => {
    // create work success handling
}).catch(() => {
    // create work failure handling
});

```


deleteCaseInCreateStage(containerItemID, ignoreCaseDeletion)

Deletes the case that is currently in the create stage.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|---|---------|--|--------------------------|
| containerItemID | string | The ID of the container item that contains information about the case that is currently in the create stage. | <input type="checkbox"/> |
| ignoreCaseDeletion | boolean | The flag that determines if a case will be deleted in the create stage. | <input type="checkbox"/> |
| <div>  NOTE: </div> | | | |

| Name | Type | Description | Required |
|------|------|---|----------|
| | | <ul style="list-style-type: none"> • The default value is <code>false</code>. • Set ignoreCaseDeletion to <code>true</code> if the case should not be deleted in the create stage. • Set ignoreCaseDeletion to <code>false</code> if the case should be deleted in the create stage. | |

Usage example

In this example, the success callback is called if the case that is currently in the create stage is deleted.

```
const deleteCaseInCreateStagePromise = getPConnect().getActionsApi().deleteCaseInCreateStage("app/modal_1");
deleteCaseInCreateStagePromise.then(() => {
  // delete case in create stage success handling
}).catch(() => {
  // delete case in create stage success handling
});
```

deleteDataObject(className, keys)

Deletes the specified data object.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------|--------|--|--------------------------|
| className | string | The name of the class that contains the data object to be deleted. | <input type="checkbox"/> |
| keys | object | The object that contains the unique identifier of the data object to be deleted. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the data object is deleted.

```
getPConnect().getActionsApi().deleteDataObject('OEDUS1-ReactApp-Data-Person', { p
yGUID: '3ba585e8-f3e2-4404-8a15-692992de53b4' }).then(() => {
  // deleteDataObject success handling
}).catch(() => {
  // deleteDataObject failure handling
});
```

eventHandler(pConn, event)

Runs an eventHandler for the given context of the component through the Action Processor.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|--------------|--------|---|--------------------------|
| pConn | object | The PConnect object associated with the handler being registered. | <input type="checkbox"/> |
| event | object | The Document Object Model (DOM) event object associated with the event. | <input type="checkbox"/> |

Usage example

In this example, the API handles the event triggered by the UI component.

```
getPConnect().getActionsApi().eventHandler(pConnObject, event);
```

fillFormWithAI(containerItemID)

Uses artificial intelligence to fill the fields in the user's assignment form with sample logical data.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerItemID | string | The ID of the container associated with the assignment form. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the form is filled successfully.



```
const fillFormWithAIPromise = getPConnect().getActionsApi().fillForWithAI('app/primary_1/workarea_1')
fillFormWithAIPromise.then(() => {
  // fillFormWithAI success handling
}).catch(() => {
  // fillFormWithAI failure handling
});
```

finishAssignment(containerItemID, options)

Submits the assignment that is associated with the given case (case details are obtained from the metadata when the application loads) and the given flow container ID.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerItemID | string | The ID of the flow container that is associated with the assignment that must be submitted. | <input type="checkbox"/> |
| options | object | The JavaScript object that contains optional properties that provide additional information for submitting the assignment. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|----------------------------|--------|--|--------------------------|
| outcomeID | string | The ID generated for an outcome of the assignment that must be submitted. | <input type="checkbox"/> |
| jsActionQueryParams | object | <p>The JavaScript object that contains properties that are added as query parameters to the fetch call of the finishAssignment API.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The jsActionQueryParams object only supports the <code>skipRoboticAutomation</code> property as of Infinity '24.2. • For more information on the <code>skipRoboticAutomation</code> property, see the query parameters of the <code>PATCH api/application/v2/cases/{caseID}/actions/{actionID}</code> endpoint. </div> | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the assignment associated with the given container information is submitted successfully.

```
const finishAssignmentPromise = getPConnect().getActionsApi().finishAssignment("app/primary_1/workarea_1");
```

```
finishAssignmentPromise.then(() => {
  // finish assignment success handling
}).catch(() => {
  // finish assignment failure handling
});
```

getDataObjectActions(dataViewID, dataViewParameters)

Obtains the available actions for a data object.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|---------------------------|--------|---|--------------------------|
| dataViewID | string | The name of the lookup data page associated with the data object. | <input type="checkbox"/> |
| dataViewParameters | object | The JavaScript object that contains the parameters required for the lookup data page. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the available actions are obtained for the data object.

```
const getDataObjectActionsPromise = getPConnect().getActionsApi().getDataObjectActions("D_DocusignEnvelopeByAccountandDocumentId", {'accountId': 'account8987', 'documentId': 'doc1234'});
getDataObjectActionsPromise.then(() => {
```

```
// getDataObjectActions success handling
}).catch() => {
  // getDataObjectActions failure handling
});
```

getDataObjectView(className, keyProperties, options)

Displays the edit view of a data object that needs to be updated.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|---|--------------------------|
| className | string | The name of the class that provides the type that the data object should belong to. | <input type="checkbox"/> |
| keyProperties | object | The object containing a set of keys to identify the data object. | <input type="checkbox"/> |
| options | object | The JavaScript object containing the property to render the edit view in the modal. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|-----------------|--------|--|--------------------------|
| viewName | string | The name of the edit view displayed for the data object. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the edit view of the data object is displayed.

```
getPConnect().getActionsApi().getDataObjectView('OZR2SN-ReactApp-Data-Person', {
  pyGUID: "61a8e531-afcb-41f7-bd72-9809e1a3cbe9"}).then(() => {
  // getDataObjectView success handling
}).catch(() => {
  // getDataObjectView failure handling
});
```


getDescendants(caseID, context)

Merges the descendants of the case in the provided context.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|---------------|--------|--|--------------------------|
| caseID | string | <p>The ID of the case whose descendants must be merged in the context.</p> <div>  NOTE: Ensure that you provide the <code>pzInsKey</code> value of the caseID. </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|----------------|--------|---|----------|
| context | string | The ID of the container item that provides the current context of the case. | ☐ |

Usage example

In this example, the success callback is called if the descendants of the `ON8TTL-C11NGALL-WORK D-206014` case are merged in the provided context.

```
const context = getPConnect().getContextName()
const getDescendantsPromise = getPConnect().getActionsApi().getDescendants("ON8TTL-C11NGALL-WORK D-206014", context)
getDescendantsPromise.then(() => {
  // getDescendants success handling
}).catch(() => {
  // getDescendants failure handling
});
```

getNextWork()

Obtains the assignment that contains the highest priority.

Returns

An empty Promise.

Parameters

This API does not have parameters.

Usage example

In this example, the success callback is called if the assignment that contains the highest priority is obtained successfully.

```
const getNextWorkPromise = getPConnect().getActionsApi().getNextWork();
getNextWorkPromise.then(() => {
  // open Next Work success handling
}).catch(() => {
  // open Next Work failure handling
});
```

getRecents(maxResultsToFetch)

Obtains the list of the recently accessed assignments.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|--------------------------|--------|---|--------------------------|
| maxResultsToFetch | number | The maximum number of results to be obtained from the server. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the specified list of the recently accessed assignments is obtained successfully.

```
const recentsPromise = getPConnect().getActionsApi().getRecents(15);
recentsPromise.then((data) => {
```

```
// success handling
}).catch(() => {
  // failure handling
});
```

ignoreDuplicateCase(containerItemID)

Performs the ignore duplicate action for the search duplicate cases automation step configured in a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|----------|
| containerItemID | string | The ID of the container item that provides the current context of the case. | ☐ |

Usage example

In this example, the API performs the ignore duplicate action for the search duplicate cases automation step configured in the `app/primary_1/workarea_1` container.

```
const ignoreDuplicateCasePromise = getPConnect().getActionsApi().ignoreDuplicateCase("app/primary_1/workarea_1");
ignoreDuplicateCasePromise.then(() => {
  // ignore duplicate case success handling
}).catch(() => {
  // ignore duplicate case failure handling
});
// The success callback is called if the ignore duplicate case action is successful.
```

invoke(url, opts)

Invokes a REST API using the fetch function of a service broker.

Returns

A Promise for the pre-processed response.

Parameters

| Name | Type | Description | Required |
|------|--------|---|--------------------------|
| url | string | The URL for the request. Example: <div><code>/prweb/api/v1/messages?filterFor=DATA-PORTAL \$SpaceTra /prweb/api/v1/data/D_pxOperatorDetails ?OperatorId=user%40c11ngallery.com</code></div> | <input type="checkbox"/> |
| opts | object | The HTTP methods that are used in the request body to fetch the REST API. | <input type="checkbox"/> |

Usage example

In this example, the API invokes the URL with the options provided, and returns a response.

```
invokePromise = getPConnect().getActionsApi().invoke(url, options);
invokePromise.then(() => {
  // invoke success handling
}).catch(() => {
```

```
// invoke failure handling
});
```

loadView(caseId, viewName, config)

Loads a view with data from server based on the context of a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|--|--------------------------|
| caseId | string | Unique identifier of the case associated with a context. | <input type="checkbox"/> |
| viewName | string | Name of the view which must be loaded with data. | <input type="checkbox"/> |
| config | object | Object that passes additional information related to the view that must be loaded. | <input type="checkbox"/> |

The following table contains the properties of the **config** object:

| Name | Type | Description | Required |
|----------------------|---------|---|--------------------------|
| containerName | string | The name of the container that the view must be loaded in. | <input type="checkbox"/> |
| context | string | The name of the context that the view must be loaded in. | <input type="checkbox"/> |
| updateData | boolean | The flag that determines the context that the view must be loaded in. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|-------------------|--------|--|----------|
| | | <p>NOTE:</p> <ul style="list-style-type: none"> • The default value of updateData is <code>false</code>. • If updateData is <code>false</code>, the view is loaded in a new context. • If updateData is <code>true</code>, the view is loaded in the specified context. | |
| actionName | string | <p>The name of the heading that must be displayed in the modal.</p> <p>NOTE: actionName must be specified only when containerName is set to <code>modal</code>.</p> | □ |

Usage example

In this example, the API loads the `pyReview` view with data from a server based on the specified context of the case whose ID is `ON8TTL-C11NGALL-WORK C-7001`.

```
const options = {
  containerName: 'preview',
  context: 'app/preview_1',
  updateData: 'true'
}
```

```
const loadViewPromise = getPConnect().getActionsApi().loadView("ON8TTL-C11NGALL-WORK C-7001","pyReview", options);
loadViewPromise.then(() => {
  // load view success handling
}).catch(() => {
  // load view failure handling
});
```

In this example, the API loads the `pyReview` view with data from a server based on the context of the case whose ID is `ON8TTL-C11NGALL-WORK C-7001`.

```
const options = {
  containerName: 'preview',
  updateData: 'false'
}
const loadViewPromise = getPConnect().getActionsApi().loadView("ON8TTL-C11NGALL-WORK C-7001","pyReview", options);
loadViewPromise.then(() => {
  // load view success handling
}).catch(() => {
  // load view failure handling
});
```

navigateToStep(stepID, containerItemID)

Navigates to a specific step in the context of a container.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| stepID | string | The ID of the step to be navigated to. Use the value <code>previous</code> to go to the previous step. | <input type="checkbox"/> |
| containerItemID | string | The ID of the container item. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the navigate action is successful.

```
navigateToStepPromise = getPConnect().getActionsApi().navigateToStep("Step1", "app/primary_1/workarea_1");
navigateToStepPromise.then(() => {
  // navigate to step success handling
}).catch(() => {
  // navigate to step failure handling
});
```

openAssignment(assignmentID, className, options)

Opens an assignment and stores it in a target container.

Supported by Pega Mobile Client™. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.


Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|---------------------|--------|---|--------------------------|
| assignmentID | string | <p>The ID of the assignment to be opened.</p> <p>Example:</p> <pre>"ASSIGN-WORKLIST OPB1HW-SPACETRA-WORK RA-43001!REQUEST_FLOW_0"</pre> | <input type="checkbox"/> |
| className | string | <p>The name of the case class associated with the assignment.</p> <p>Example:</p> <pre>"OPB1HW-SpaceTra-Work-RequestApproval"</pre> | <input type="checkbox"/> |
| options | object | The JavaScript object containing a property of the target container. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|----------------------|--------|---|--------------------------|
| containerName | string | <p>The name of the target container that will store the assignment.</p> <div>  NOTE: The default value of this property is <code>primary</code>. </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|-----------------------|--------|--|----------|
| | | Example: <div>{containerName: "primary"}</div> | |
| remoteCaseMeta | object | The JSON object containing starting fields to be set for opening a traditional remote case assignment. | ☐ |

Usage example

In this example, the success callback is called if the assignment is opened successfully.

```
const openAssignmentPromise = getPConnect().getActionsApi().openAssignment(
  "ASSIGN-WORKLIST OPB1HW-SPACETRA-WORK RA-43001!REQUEST_FLOW_0",
  "OPB1HW-SpaceTra-Work-RequestApproval",
  { containerName: "primary" }
);
openAssignmentPromise.then(() => {
  // open assignment success handling
}).catch(() => {
  // open assignment failure handling
});
```

openDataObjectAction(className, keyProperties, actionID, actionName)

Displays the flow action view of the data object to be updated.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|--------------------------|
| className | string | The class ID of the object's data type. | <input type="checkbox"/> |
| keyProperties | object | The object containing values of the data object. | <input type="checkbox"/> |
| actionID | string | The ID of the action that is configured on the object's data type. | <input type="checkbox"/> |
| actionName | string | The name of the action that is configured on the object's data type. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the flow action view of the specified data object is displayed.

```
const openDataObjectActionPromise = getPConnect().getActionsApi().openDataObjectAction("ON8TTL-C11nGall-Data-Card", {Number:1234}, 'updateCVV', 'Update CVV');
openDataObjectActionPromise.then(() => {
  openDataObjectAction success handling
}).catch(() => {
  openDataObjectAction failure handling
});
```

openEmbeddedDataModal(viewName, component, targetProperty, index, action, heading)

Displays the view to add or update a record in the Embedded data type.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|-----------------------|--------|---|--------------------------|
| viewName | string | The name of the view that is displayed in the modal. | <input type="checkbox"/> |
| component | object | The PConnect object associated with the Embedded data table. | <input type="checkbox"/> |
| targetProperty | string | The property associated with the Embedded data type. | <input type="checkbox"/> |
| index | number | The position of the row within the Embedded data table in which the record will be added or updated. | <input type="checkbox"/> |
| action | string | The action to be performed on the record. <ul style="list-style-type: none"> • CREATE - Adds the record to the Embedded data type. • EDIT - Updates the record in the Embedded data type. | <input type="checkbox"/> |
| heading | string | The heading that is displayed in the modal based on the action performed on the record. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the modal is opened successfully.

```
getPConnect().getActionsApi().openEmbeddedDataModal('Create', pConnObject, '.Employees', 1, 'CREATE', 'Add Employee').then(() => {
  // openEmbeddedDataModal success handling
}).catch(() => {
```

```
// openEmbeddedDataModal failure handling
})
```

openLocalAction(actionID, options)

Opens the configured local action.

Supported in Pega Mobile Client™ release 12.1.0 or later. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.


Returns

A Promise associated with the action.

Parameters


| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| actionID | string | The unique identifier of the local action. | <input type="checkbox"/> |
| options | object | The JavaScript object containing the properties for opening the local action. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|---|--------|---|--------------------------|
| caseID | string | The unique ID of the case. If this ID is not provided, it will be obtained using the current context. | <input type="checkbox"/> |
| <div>  NOTE: </div> | | | |

| Name | Type | Description | Required |
|------------------|--------|---|--------------------------|
| | | <ul style="list-style-type: none"> This is applicable for <code>Case / Stage</code> wide local action. Ensure that you provide the <code>pzInsKey</code> value of the caseID. | |
| assignKey | string | <p>The unique ID of the assignment. If this ID is not provided, it will be obtained using the current context.</p> <p>NOTE: This is applicable for local action of type <code>Assignment</code>.</p> | <input type="checkbox"/> |
| name | string | The title of the local action. This is used as the modal header title when the local action is displayed in the modal dialog. | <input type="checkbox"/> |
| type | string | <p>The type of the local action. The values are:</p> <ul style="list-style-type: none"> <code>Assignment</code> - Used for an assignment level local action. <p>Example:</p> <pre>{ "name": "Transfer assignment", "type": "Assignment",</pre> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <div><pre>"containerName": "modal" }</pre></div> <ul style="list-style-type: none">Case - Used for a case-wide local action. <p>Example:</p> <div><pre>{ "name": "Change stage", "type": "Case", "containerName": "workarea" }</pre></div> <ul style="list-style-type: none">Express - Used to call the case-wide local action from a different page context other than the case page. <p>Example:</p> <div><pre>{ "name": "Edit details", "type": "express", "containerName": "modal", "assignKey": "ASSIGN-WORKLIST ON 8TTL-C11NGALL-WORK D-206014!VE HICLEDDETAILS_FLOW", "caseID": "ON8TTL-C11NGALL-WORK D-206014" }</pre></div> | |

| Name | Type | Description | Required |
|--------------------------|--------|---|--------------------------|
| | | <ul style="list-style-type: none"> • Stage - Used for a stage-wide local action. <p>Example:</p> <pre>{ "name": "Edit Details", "type": "stage", "containerName": "modal" }</pre> | |
| containerName | string | The name of the container that is used to launch the local action in a modal or a work area. | <input type="checkbox"/> |
| actionTitle | string | The title of the local action. This is displayed as the title of the modal header when the local action is displayed in a modal dialog. | <input type="checkbox"/> |
| refreshConditions | array | <p>An array of objects containing the field name and event for use in performing a conditional refresh in an opened local action.</p> <div>  NOTE: This is applicable only for express type local actions. </div> | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the local action is launched successfully.

```
const localActionPromise = getPConnect().getActionsApi().openLocalAction("EditDetails",
{
  type:"Express",
  containerName:"modal",
  assignKey:"ASSIGN-WORKLIST ON8TTL-C11NGALL-WORK D-206014!VEHICLEDETAILS_FLOW",
  caseID:"ON8TTL-C11NGALL-WORK D-206014",
  refreshConditions:[
    {
      field:".Prop1",
      event:"Changes"
    }
  ]
});
localActionPromise.then(() =>{
  // local action success handling
}).catch(() =>{
  // local action failure handling
});
```

openProcessAction(actionID, options)

Opens the configured process action.

Supported in Pega Mobile Client™ release 12.1.0 or later. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.

Returns


A Promise associated with the action.



Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| actionID | string | The ID of the process action. | <input type="checkbox"/> |
| options | object | The JSON object containing the container target in which the process action is displayed. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|--------------------|--------|--|--------------------------|
| caseID | string | <p>The unique ID of the case. If this ID is not provided, it will be obtained using the current context.</p> <div>  NOTE: Ensure that you provide the <code>pzInsKey</code> value of the caseID. </div> | <input type="checkbox"/> |
| actionTitle | string | The title of the process action. This is used as the modal header title when the process action is displayed in the modal dialog. | <input type="checkbox"/> |
| type | string | <p>The type of the process action. The values are <code>Case</code> or <code>Stage</code>.</p> <p>Example:</p> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|---|----------|
| | | <code>{caseID: "", actionTitle: "Vendor Addition", type: "Case" / "Stage"}</code> | |

Usage example

In this example, the success callback is called if the process action is launched successfully.

```
const processActionPromise = getPConnect().getActionsApi().openProcessAction("VendorAddition_Flow", { type: "Case" });
processActionPromise.then(() => {
  // open process action success handling
}).catch(() => {
  // open process action failure handling
});
```

openStage(stageID)

Opens the specified stage associated with a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------|--------|---|----------|
| stageID | string | The ID of the stage associated with a case. | ☐ |

Usage example

In this example, the success callback is called if the `PRIM3` stage is opened successfully.

```
const openStagePromise = getPConnect().getActionsApi().openStage("PRIM3");
openStagePromise.then(() => {
  // open stage success handling
}).catch(() => {
  // open stage failure handling
});
```

openWorkByHandle(workID, className, options)

Opens a work object associated with a work ID.

Supported by Pega Mobile Client™. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------|--------|---|----------|
| workID | string | The work object to be opened. | ☐ |
| className | string | The name of the case class associated with the work object. | ☐ |

| Name | Type | Description | Required |
|----------------|--------|--|--------------------------|
| options | object | The JavaScript object that contains the properties required for opening the work object. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|------------------------|--------|---|--------------------------|
| targetContainer | string | The name of the container that the work object will be opened in. | <input type="checkbox"/> |
| remoteCaseMeta | object | The JSON object containing starting fields to be set for opening a traditional remote case. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the work object is opened successfully.

```
const openWorkPromise = getPConnect().getActionsApi().openWorkByHandle("OPB1
HW-SPACETRA-WORK RA-10001", "OPB1HW-SpaceTra-Work-RequestApproval");
openWorkPromise.then(() => {
  // open work by handle success handling
}).catch(() => {
  // open work by handle failure handling
});
```

rejectCase(containerItemID)

Performs the reject action for the approval step configured in a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|-------------------------------|----------|
| containerItemID | object | The ID of the flow container. | ☐ |

Usage example

In this example, the success callback is called if the reject action is successful.

```
const rejectCasePromise = getPConnect().getActionsApi().rejectCase("app/primary_1/workarea_1");
rejectCasePromise.then(() => {
  // reject case success handling
}).catch(() => {
  // reject case failure handling
});
```

resolveDuplicateCase(containerItemID)

Performs the resolve duplicate action for the search duplicate cases automation step configured in a case.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|----------|
| containerItemID | string | The ID of the container item that provides the current context of the case. | ☐ |

Usage example

In this example, the API performs the resolve duplicate action for the search duplicate cases automation step configured in the `app/primary_1/workarea_1` container.

```
const resolveDuplicateCasePromise = getPConnect().getActionsApi().resolveDuplicateCase("app/primary_1/workarea_1");
resolveDuplicateCasePromise.then(() => {
  // resolve duplicate case success handling
}).catch(() => {
  // resolve duplicate case failure handling
});
// The success callback is called if the resolve duplicate case action is successful.
```

saveAndClose(containerItemID)

Saves the current work object opened in the modal dialog.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerItemID | string | The ID of the container item that contains information about the current work object. Example: <code>"app/modal_1"</code> | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the case is saved successfully.

```
const saveAndClosePromise = getPConnect().getActionsApi().saveAndClose("app/modal_1");
saveAndClosePromise.then(() => {
  // save and close success handling
}).catch(() => {
  // save and close failure handling
});
```

saveAssignment(containerItemID)

Saves the opened assignment.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerItemID | string | The ID of the container item that contains information about the opened assignment. Example: <code>"app/modal_1"</code> | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the opened assignment is saved successfully.

```
const saveAssignmentPromise = getPConnect().getActionsApi().saveAssignment("app/modal_1");
saveAssignmentPromise.then(() => {
  // saveAssignment success handling
}).catch(() => {
  // saveAssignment failure handling
});
```

showCasePreview(pzInsKey, configObj)

Displays a preview of a case item inside the preview panel.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|------------------|--------|--|--------------------------|
| pzInsKey | string | The unique identifier of the case item. Example: 0PB1HW-SPACETRA-WORK 20RA-2 | <input type="checkbox"/> |
| configObj | object | The name of the case item's class. | <input type="checkbox"/> |

Usage example

In this example, the API displays the preview of a case item (identified by a unique key called `pzInsKey`) inside the preview panel.

```
getPConnect().getActionsApi().showCasePreview(pzInsKey, configObj);
```

showData(pageName, dataContext, dataContextParameters, options)

Displays the data of a page based on the data context.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| pageName | string | The name of the view in which the data must be displayed. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------------------------------|--------|--|--------------------------|
| dataContext | string | The name of the page type data page whose data must be displayed. | <input type="checkbox"/> |
| dataContextParameters | object | The parameters associated with the data page. | <input type="checkbox"/> |
| options | object | The JavaScript object containing the properties to display the data. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|--|--------------------------|
| containerName | string | The name of the container that displays the data. | <input type="checkbox"/> |
| skipSemanticUrl | boolean | <p>The flag that determines if the semantic URL evaluation logic must be skipped.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipSemanticUrl is <code>false</code>, the semantic URL evaluation logic will not be skipped. • If skipSemanticUrl is <code>true</code>, the semantic URL evaluation logic will be skipped. </div> | <input type="checkbox"/> |

Usage example

In this example, the API displays the data of the `EmployeeDetails` page based on the `D_EmployeeDetails` data context.

```
const showDataPromise = getPConnect().getActionsApi().showData("EmployeeDetails", "D_EmployeeDetails", {pyGUID: "0759409f-4146-439c-aa25-57d4f495fee5"});
showDataPromise.then(() => {
  // show data success handling
}).catch(() => {
  // show data failure handling
});
```

showDataObjectCreateView(className, viewName)

Displays the create view of a data object to be created.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------|--------|---|--------------------------|
| className | string | The name of the class that provides the type that the data object should belong to. | <input type="checkbox"/> |
| viewName | string | The name of the create view that is to be displayed for the data object. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the create view is displayed.

```
getPConnect().getActionsApi().showDataObjectCreateView('OZR2SN-ReactApp-Data-Person', 'Create').then(() => {
// showDataObjectCreateView success handling
}).catch(() => {
// showDataObjectCreateView failure handling
});
```

showDataPreview(dataContext, dataContextParameters, options)

Displays a preview of a data item in the preview panel.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------------|--------|--|--------------------------|
| dataContext | string | The name of the data page whose data must be displayed. | <input type="checkbox"/> |
| dataContextParameters | string | The parameters associated with the data page. | <input type="checkbox"/> |
| options | object | The object containing properties required to perform additional actions while opening the preview panel. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|---|--------------------------|
| container | string | The name of the container. | <input type="checkbox"/> |
| containerName | string | The name of the container that displays the preview of the data. | <input type="checkbox"/> |
| context | string | The context in which the preview of the data is displayed. | <input type="checkbox"/> |
| skipSemanticUrl | boolean | <p>The flag that determines if the semantic URL evaluation logic must be skipped. The semantic URL evaluation logic is used to check if the current data preview is available, which determines if the preview container is activated or opened.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipSemanticUrl is <code>true</code>, the semantic URL evaluation logic is skipped and the preview container is always open. • If skipSemanticUrl is <code>false</code>, the semantic URL evaluation logic is applied and the preview container is activated. </div> | <input type="checkbox"/> |

Usage example

In this example, the API displays the preview of a data item (identified by the `D_TestData` data page and the `ProductId` parameter having the value `prd-1`) inside the preview panel with the extra configuration object provided.

```
getPConnect().getActionsApi().showDataPreview('D_TestData', {ProductId : 'prd-1'});
```

showPage(pageName, className, options)

Displays the page of a class.

Supported in Pega Mobile Client™ release 11.3.0 or later. When Pega Mobile Client invokes this API, it creates a new native screen or a new separate WebView component. As a result, you cannot access or modify the internal state of the resulting screen from the `then` block of the Promise.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------|--------|--|--------------------------|
| pageName | string | The name of the page to be displayed. Example: <code>pageName = "pyHome"</code> | <input type="checkbox"/> |
| className | string | The name of the class to which the page belongs. Example: <code>className: "Data-Portal"</code> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|----------------|--------|--|--------------------------|
| options | object | The JavaScript object containing the properties to display the page. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerName | string | The name of the container that displays the page. | <input type="checkbox"/> |
| skipSemanticUrl | string | <p>The flag that determines if the semantic URL evaluation logic must be skipped.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipSemanticUrl is <code>false</code>, the semantic URL evaluation logic will not be skipped. • If skipSemanticUrl is <code>true</code>, the semantic URL evaluation logic will be skipped. </div> | <input type="checkbox"/> |

Usage example

In this example, the API obtains and displays the view metadata and class name of a page.

```
const showPagePromise = getPConnect().getActionsApi().showPage("pyHome", "Data-Portal");
showPagePromise.then(() => {
  // show page success handling
}).catch(() => {
  // show page failure handling
});
```

submitDataObjectAction(containerItemID, keyProperties, actionID)

Updates the data object based on the flow action.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|--------------------------|
| containerItemID | string | The ID of the container item containing the edit view of the data object. | <input type="checkbox"/> |
| keyProperties | object | The object containing the keys of the data object to be updated. | <input type="checkbox"/> |
| actionID | string | The ID of the action that is configured on the object's data type. | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the specified data object is updated based on the `updateCVV` flow action.

```
const submitDataObjectActionPromise = getPConnect().getActionsApi().submitDataObjectAction("app/primary_1/modal_1", {Number:1234}, 'updateCVV');
submitDataObjectActionPromise.then(() => {
  submitDataObjectAction success handling
}).catch(() => {
  submitDataObjectAction failure handling
});
```

submitEmbeddedDataModal(containerItemID)

Submits a view and adds or edits a record in the Embedded data type.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|---|----------|
| containerItemID | string | The ID of the container item containing the add or edit view of the Embedded data type. | ☐ |

Usage example

In this example, the success callback is called if a row is added or edited successfully.

```
getPConnect().getActionsApi().submitEmbeddedDataModal('app/modal_3').then(() => {
  // submitEmbeddedDataModal success handling
}).catch(() => {
  // submitEmbeddedDataModal failure handling
})
```

triggerFieldChange(propName, value, skipValidation)

Runs validations on a specified control and triggers the `FIELD_CHANGE` event.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------------|---------|---|--------------------------|
| propName | string | The name of the control on which the <code>FIELD_CHANGE</code> event must be invoked. | <input type="checkbox"/> |
| value | any | The value of the control that will be validated and passed when invoking the <code>FIELD_CHANGE</code> event. | <input type="checkbox"/> |
| skipValidation | boolean | <p>The flag that determines if client-side validation must be run.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • Set skipValidation to <code>true</code> if client-side validation must be skipped. • Set skipValidation to <code>false</code> if client-side validation must be run. </div> | <input type="checkbox"/> |

Usage example

In this example, the API validates and invokes the `FIELD_CHANGE` event as triggered by the specified control.

```
getPConnect().getActionsApi().triggerFieldChange(propName, value);
```

updateDataObject(containerItemID, keys)

Updates a data object whose edit view is loaded in the specified container item. This update is made in the database.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|--------------------------|
| containerItemID | string | The ID of the container item containing the edit view of the data object. | <input type="checkbox"/> |
| keys | object | The keys of the data record to be updated. Example: <pre>{ "pyGUID": "b1715d32-7d7b-4d41-97ce-ce d87b7c8fb0" }</pre> | <input type="checkbox"/> |

Usage example

In this example, the success callback is called if the data object is updated.

```
const updateDataObjectPromise = getPConnect().getActionsApi().updateDataObject("
app/modal_3", {"pyGUID":"61a8e531-afcb-41f7-bd72-9809e1a3cbe9"});
updateDataObjectPromise.then(() => {
  updateDataObject success handling
}).catch(() => {
  updateDataObject failure handling
});
```

updateFieldValue(propName, value, options)

Dispatches the `SET_PROPERTY` event on a specified control to store a specified value in the Redux Store.

This API also deletes the error messages of the control by calling the [clearMessages\(config\)](#) API in the MessageManager class.

Returns


Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|----------|
| propName | string | The name of the control whose value must be stored in the Redux Store. | ☐ |
| value | any | The value of the control to be stored in the Redux Store. | ☐ |
| options | object | The JavaScript object containing the properties to enhance the functionality of this API. | ☐ |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|--------------------------------------|---------|---|--------------------------|
| removePropertyFromChangedList | boolean | <p>The flag that removes the entry of the property from the <code>changedPropertyList</code>.</p> <div> <p>NOTE: The default value is <code>false</code>. Set this property to <code>true</code> to remove the entry of the property from the <code>changedPropertyList</code>.</p> </div> | <input type="checkbox"/> |
| isArrayDeepMerge | boolean | <p>The flag that determines if the values within an array or object must be deep merged to the Redux Store.</p> <div> <p>NOTE: The default value is <code>true</code>. Set this property to <code>true</code> to deep merge the values to the Redux Store.</p> </div> | <input type="checkbox"/> |
| skipDirtyValidation | boolean | <p>The flag that determines if the property must be validated.</p> <div> <p>NOTE: The default value is <code>false</code>. Set this property to <code>true</code> to skip validating the propName property.</p> </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|--------------------|---------|---|--------------------------|
| isListEntry | boolean | <p>The flag that determines if the error related data for a specific entry within a list should be deleted.</p> <div>  <p>NOTE: The default value is <code>false</code>. Set this property to <code>true</code> to delete all error related data for a specific entry within a list.</p> </div> | <input type="checkbox"/> |

Usage example

In this example, the API stores the value passed in the Redux Store as triggered by the control.

```
getPConnect().getActionsApi().updateFieldValue(propName, value);
```

APIs in the CaseInfo class

Use the APIs in the CaseInfo class to handle the information related to a case.

- `getActions()`
- `getActiveFlowActionID()`
- `getAssignmentID()`
- `getAvailableProcesses()`
- `getBusinessID()`
- `getCaseStages(caseID, context)`

- `getCaseTypeName()`
- `getClassName()`
- `getCurrentAssignmentViewName()`
- `getID()`
- `getKey()`
- `getName()`
- `getParentCaseInfo()`
- `isPerform()`
- `isReview()`

`getActions()`

Obtains information related to case actions that can be performed on an assignment.

Returns

Information related to case actions as an array.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns information related to case actions.

```
const actions = getPConnect().getCaseInfo().getActions();
```

getActiveFlowActionID()

Obtains the flow action ID of the rendered assignment.

Returns

The flow action ID as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the rendered flow action ID.

```
const activeFlowActionID = getPConnect().getCaseInfo().getActiveFlowActionID();
```

getAssignmentID()

Obtains the ID of the assignment that is rendered using the current context.

Returns

The ID of the assignment as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ID of the assignment that is rendered using the current context.

```
const assignmentID = getPConnect().getCaseInfo().getAssignmentID();
```

getAvailableProcesses()

Obtains the available optional processes for a case.

Returns

The available optional processes as an array.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the available optional processes for a case.

```
const availableProcesses = getPConnect().getCaseInfo().getAvailableProcesses();
```

The available optional processes are returned as shown in the array below:

```
[{
  "name": "Vendor Addition",
  "links": {
    "add": {
      "rel": "self",
      "href": "/cases/ON8TTL-C11NGALL-WORK PC-99001/processes/VendorAddition_
Flow",
      "type": "POST",
      "title": "add case optional process"
    }
  },
  "ID": "VendorAddition_Flow",
  "type": "Case"
}]
```

getBusinessID()

Obtains the business ID of a case that is rendered using the current context.

Returns

The business ID as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns `SR-71086` as the business ID of the case.

```
const caseKey = getPConnect().getCaseInfo().getBusinessID();
```

getCaseStages(caseID, context)

Obtains the details of the case stages for a specified case.

Returns

A promise that resolves to an object containing the details of the case stages.

For more information on the response body containing the details of the case stages, see [Successful response](#).

Parameters

| Name | Type | Description | Required |
|--------|--------|---|----------|
| caseID | string | The unique identifier of the case whose case stage details must be fetched. | ☐ |

| Name | Type | Description | Required |
|----------------|--------|---|----------|
| context | string | The name of the context where the details of the case stages for a specified case are rendered. | ☐ |

Usage example

In this example, the API returns the details of the case stages for the case with ID `OPB1HW-SPACETRA-WORK RA-43001` rendered in the `app/primary_1` context.

```
getPConnect()
  .getCaseInfo()
  .getCaseStages('OPB1HW-SPACETRA-WORK RA-43001', 'app/primary_1')
  .then((stages) => {
    // success handling
    console.log(stages);
  })
  .catch((error) => {
    // failure handling
  });
```

getCaseTypeName()

Obtains the name of the case type.

Returns

The name of the case type as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the case type.

```
const caseName = getPConnect().getCaseInfo().getCaseTypeName();
```

getClassName()

Obtains the name of the case class.

Returns

The name of the case class as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the case class.

```
const className = getPConnect().getCaseInfo().getClassName();
```

getCurrentAssignmentViewName()

Obtains the name of the current assignment's view in a case.

Returns

The name of the current assignment's view as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the name of the current assignment's view in a case.

```
const viewName = getPConnect().getCaseInfo().getCurrentAssignmentViewName();
```

getID()

Obtains the ID of a case that is rendered using the current context.

Returns

The ID as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ID of a case that is rendered using the current context.

```
const caseKey = getPConnect().getCaseInfo().getID();
```

getKey()

Obtains the key of the case that is rendered using the current context.

Returns

The case key as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the `ON8TTL-C11NGALL-WORK CST-1` key of the case.

```
const caseKey = getPConnect().getCaseInfo().getKey();
```

getName()

Obtains the name of the case that is rendered using the current context.

Returns

The name of the case as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the case that is rendered using the current context.

```
const caseName = getPConnect().getCaseInfo().getName();
```

getParentCaseInfo()

Obtains the information of the parent case when the current rendered assignment belongs to a child case.

Returns

The parent case details of the current child case as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the information of the parent case when the current rendered assignment belongs to a child case.

```
const parentCaseInfo = getPConnect().getCaseInfo().getParentCaseInfo();
```

isPerform()

Determines if the view is in the perform mode.

Returns

The Boolean value `true` if the view is in the perform mode.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the Boolean value `true` if the view is in the perform mode.

```
const isPerform = getPConnect().getCaseInfo().isPerform();
```

isReview()

Determines if the view is in the review mode.

Returns

The Boolean value `true` if the view is in the review mode.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the Boolean value `true` if the view is in the review mode.

```
const isReview = getPConnect().getCaseInfo().isReview();
```

APIs in the ContainerManager class

Use the APIs in the ContainerManager class to manage the life cycle of a container.

For more information on containers, see [Working with Containers](#).

- [activateContainerItem\(containerInfo\)](#)
- [addContainerItem\(containerInfo\)](#)
- [addTransientItem\(containerInfo\)](#)
- [initializeContainers\(containerInfo\)](#)
- [removeContainerItem\(containerInfo\)](#)
- [removeTransientItem\(transientItemInfo\)](#)
- [resetContainers\(containerInfo\)](#)
- [updateContainerItem\(containerInfo\)](#)

- [updateTransientData\(transientObject\)](#)

activateContainerItem(containerInfo)

Activates an item in a container.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|----------|
| containerInfo | object | The JSON object containing information about the container item to be activated. | ☐ |

Usage example

In this example, the API activates the specified item in the container.

```
const containerManager = getPConnect().getContainerManager(); containerManager.  
activateContainerItem({ target: "app/primary", containerItemID: "app/primary_3" });
```

addContainerItem(containerInfo)

Adds an item to a container.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|--------------------------|
| containerInfo | object | <p>The JSON object containing information about the semantic URL associated with the context of the container.</p> <p>You can specify additional information such as the targeted region in the container (<code>acName</code>) and the way the information should be rendered in the container (<code>showList</code>).</p> <p>Example:</p> <pre>{ semanticURL: "", context: "app", acName : "primary" } { semanticURL: "", context: "app/primary_5", showList: false }</pre> | <input type="checkbox"/> |

Usage example

In this example, the API adds an item to a Container.

```
const containerManager = getPConnect().getContainerManager();
containerManager.addContainerItem({
  context: 'app/primary_1',
  semanticURL: "RequestApprovals/REQ-1",
  showList: true
});
```

addTransientItem(containerInfo)

Adds a transient item to the current context. The transient item contains values that have not been submitted with the content.

Returns

The newly created transient item ID as a string.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|---|--------------------------|
| containerInfo | object | The JSON object containing information about the unique ID of the transient item and the data containing the properties and their values. | <input type="checkbox"/> |

Usage example

In this example, the API adds a transient item along with its data to the current context.

```
const containerManager = getPConnect().getContainerManager();
containerManager.addTransientItem({
  id: 'uniqueIdentifier',
  data: {
    "Prop1": "valueA",
    "Prop2": "valueB"
  }
});
```

initializeContainers(containerInfo)

Initializes containers in a complex component.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|---|--------------------------|
| containerInfo | object | The type of the container being initialized. Example: <div><pre>{type: "single"} {type: "multiple"}</pre></div> | <input type="checkbox"/> |

Usage example

In this example, the API initializes a container in the Store.

```
const containerManager = getPConnect().getContainerManager();  
containerManager.initializeContainers({  
  type: "multiple"  
});
```

removeContainerItem(containerInfo)

Removes an item from a container.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|----------|
| containerInfo | object | The JSON object containing information about the container item to be removed. | ☐ |

Usage example

In this example, the API removes the specified item from the container.

```
const containerManager = getPConnect().getContainerManager();
containerManager.removeContainerItem({ target: "app/primary", containerItemID: "app/primary_3" });
```

removeTransientItem(transientItemInfo)

Removes a transient item from the current context.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|--------------------------|--------|---|----------|
| transientItemInfo | object | The JSON object containing the target, the ID of the container, and the ID of the transient item. | ☐ |

Usage example

In this example, the API removes a transient item from the current context.

```
const containerManager = getPConnect().getContainerManager();
containerManager.removeTransientItem({
  target:'uniqueTarget',
  containerItemID:'uniqueContainerID',
  transientItemID:'uniqueTransientID',
});
```

resetContainers(containerInfo)

Resets a container to its initial state.

Returns

A promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|--------------------------|
| containerInfo | object | The object containing information about the container being reset. | <input type="checkbox"/> |

Usage example

In this example, the API resets the container to its initial state.

```
const containerManager = getPConnect().getContainerManager();
containerManager.resetContainers({
  context:"app",
  name:"preview",
  containerItems: ["app/preview_1","app/preview_2"]
});
```


updateContainerItem(containerInfo)

Updates the information within an item in the container.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|----------------------|--------|--|--------------------------|
| containerInfo | object | <p>The JSON object containing information about the context, target, container item ID, and any semantic URL associated with the context of the container.</p> <p>You can specify additional information such as the targeted region in the container (<code>acName</code>) and the way the information should be rendered in the container (<code>showList</code>).</p> | <input type="checkbox"/> |

Usage example

In this example, the API updates the information within an item in the container.

```
const containerManager = getPConnect().getContainerManager();
containerManager.updateContainerItem({
  semanticURL: "RequestApprovals/REQ-1",
  caseViewMode: "review",
  target:"app/primary_0/workarea",
  containerItemID:"app/primary_0/workarea_0",
```

```
context:"app/primary_0",
});
```

updateTransientData(transientObject)

Updates a transient item in the current context, which can be used to hold the values that don't get submitted with the content.

Returns

A Promise associated with the action.

Parameters

| Name | Type | Description | Required |
|------------------------|--------|--|----------|
| transientObject | object | The JSON object containing the ID of the transient item and data containing the properties and their values. | ☐ |

Usage example

In this example, the API updates a transient item in the current context.

```
const containerManager = getPConnect().getContainerManager();
containerManager.updateTransientData({
  transientItemID: 'uniqueIdentifier',
  data: {
    "Prop1": "valueA",
    "Prop2": "valueB"
  }
});
```

APIs in the ListActions class

Use the APIs in the ListActions class to handle list related actions for components that display page list data and allow modifications on the list.

For more information on page instructions, see [Page instructions for page lists](#) .

- **`clearSelectedRows()`**
- **`deleteAll(property, options)`**
- **`deleteEntry(index, pageRef)`**
- **`deletePage(property)`**
- **`getSelectedRows(withPayload)`**
- **`initDefaultPageInstructions(property, propertyNames, uniqueField)`**
- **`insert(payload, index, pageRef, options)`**
- **`reorder(fromIndex, toIndex)`**
- **`replacePage(property, payload)`**
- **`setSelectedRows(rows, options)`**
- **`setVisibility(isVisible)`**
- **`update(payload, index, options)`**
- **`updateProperty(propertyName, value, options)`**

clearSelectedRows()

Deletes all the previously selected rows by adding the DELETE ALL page instruction to the case or data reference list and updates the Redux Store.

Returns

Not applicable.

Parameters

This API does not have parameters.

Usage example

In this example, the API deletes all the selected rows for the list page and inserts the DELETE ALL instruction for the submit action.

```
getPConnect().getListActions().clearSelectedRows()
```

The empty `Employees` list in Redux is as shown below:

```
Employees: []
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [{
  "target": ".Employees",
  "content": {
    pyGUID: '72def3c5-3eda-42cc-b06a-c92fab6feb16'
  },
  "listIndex": 1,
  "instruction": "INSERT"
},
{
  "target": ".Employees",
  "content": {
    pyGUID: '72def3c5-3eda-42cc-b06a-c92fab6feb15'
```

```

    },
    "listIndex": 2,
    "instruction": "INSERT"
  },
  {
    "target": ".Employees",
    "instruction": "DELETE ALL"
  }
]

```

deleteAll(property, options)

Adds the DELETE_ALL page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| property | string | The unique property of the list to which the DELETE_ALL page instruction must be added. | <input type="checkbox"/> |
| options | object | The object containing additional information to perform the DELETE_ALL operation. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|--|--------------------------|
| skipStateUpdate | boolean | The flag that determines if the Redux state must be updated. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | <div> <div>NOTE:</div> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If <code>skipStateUpdate</code> is <code>true</code>, the Redux state will be updated. • If <code>skipStateUpdate</code> is <code>true</code>, the Redux state will not be updated. </div> | |

Usage example

In this example, the API adds the DELETE_ALL page instruction to the `ShippingAddress` property and does not update the Redux state.

```
getPConnect().getListActions().deleteAll('.ShippingAddress', { skipStateUpdate : false }
);
```

deleteEntry(index, pageRef)

Adds the DELETE page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|----------------|--------|---|--------------------------|
| index | number | The position in the list that must be deleted. | <input type="checkbox"/> |
| pageRef | string | The reference to the page within the current context. | <input type="checkbox"/> |

Usage example

In this example, the API deletes the specified row in the referenced list page and inserts the DELETE instruction for the submit action.

```
getPConnect().getListActions().deleteEntry(0, 'caseInfo.content')
```

The deleted `Name` property in the `EmployeeDetails` list in Redux is as shown below:

```
EmployeeDetails: []
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [{
  target: ".EmployeeDetails",
  content: {},
  listIndex: 1,
  instruction: "INSERT"
},
{
  target: ".EmployeeDetails",
  content: {
    Name: 'John'
```

```

    },
    listIndex: 1,
    instruction: "UPDATE"
  },
  {
    target: ".EmployeeDetails",
    listIndex: 1,
    instruction: "DELETE"
  }
]

```

deletePage(property)

Adds the DELETE page instruction for a specific field in a row of the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|--|----------|
| property | string | The name of the property in a field whose value must be deleted. | ☐ |

Usage example

In this example, the API deletes the data present in the `addressProof` property in a row of the referenced list page.

```
getPConnect().getListActions().deletePage('.addressProof')
```


getSelectedRows(withPayload)

Obtains the data for selected rows from the referenced list in the context.

Returns

The row data as an array of objects.

Parameters

| Name | Type | Description | Required |
|--------------------|---------|---|--------------------------|
| withPayload | boolean | <p>The flag that determines the scope of the data in the selected rows that must be included in the response.</p> <ul style="list-style-type: none">• The default value of withPayload is <code>false</code>.• Set withPayload to <code>true</code> to include all the data of the selected rows in the response.• Set withPayload to <code>false</code> to include the key properties of the selected rows in the response. | <input type="checkbox"/> |

Usage example

In this example, the API returns the key properties of the selected rows as an array.

```
getPConnect().getListActions().getSelectedRows(false)
```

The array containing the key properties of the selected rows is shown below:

```
[  
  { pyGUID : "1f8cce17-619d-41e9-94a3-179426eb5569"  
  },  
  { pyGUID : "02517df0-f460-4673-b873-2fea889af6d2"  
  }  
]
```

initDefaultPageInstructions(property, propertyNames, uniqueField)

Adds the UPDATE page instruction to the rows populated by default of the referenced list in the context.



NOTE: Use this API when the list has existing rows whose data needs to be passed in the payload of the submit action, and after the last entry in the list is mounted.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|----------------------|----------|---|--------------------------|
| property | string | The name of the referenced list to which the UPDATE page instruction must be added. | <input type="checkbox"/> |
| propertyNames | string[] | The list of fields of the referenced list to which the UPDATE page instruction must be added. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|--------------------|--------|---|----------|
| uniqueField | string | The unique field of a row to which the UPDATE page instruction must be added. | ☐ |

Usage example

In this example, the API adds the UPDATE page instruction to the rows populated by default of the `Employees` list for the `pyGUID` and `pyName` fields that are uniquely identified by the value of the `EmbedListUUID__` field.

```
getPConnect().getListActions().initDefaultPageInstructions('.Employees', ['pyGUID', 'pyName'], '.EmbedListUUID__')
```

insert(payload, index, pageRef, options)

Adds the INSERT page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|----------------|--------|---|----------|
| payload | object | The object containing the values to be inserted in the referenced list. | ☐ |
| index | number | The position that must be inserted in the referenced list. | ☐ |
| pageRef | string | The reference to the page within the current context. | ☐ |

| Name | Type | Description | Required |
|----------------|--------|---|--------------------------|
| options | object | The object containing additional information to perform the INSERT operation. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|--|--------------------------|
| skipStateUpdate | boolean | <p>The flag that determines if the Redux State must be updated.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipStateUpdate is <code>true</code>, the Redux State is not updated. • If skipStateUpdate is <code>false</code>, the Redux State is updated. </div> | <input type="checkbox"/> |

Usage example

In this example, the API inserts a row for list page and inserts an INSERT instruction for the submit action.

```
getPConnect().getListActions().insert({}, 0, 'caseInfo.content', {
  skipStateUpdate: false
})
```

The inserted row in the `EmployeeDetails` list in Redux is as shown below:

```
EmployeeDetails: [{
  classID: 'ON8TTL-C11nGall-Data...ableData'
}]
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [{
  "target": ".EmployeeDetails",
  "content": {},
  "listIndex": 1,
  "instruction": "INSERT"
}]
```

reorder(fromIndex, toIndex)

Adds the MOVE page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|------------------|--------|--|--------------------------|
| fromIndex | number | The position in the list from which the row must be moved. | <input type="checkbox"/> |
| toIndex | number | The position in the list to which the row must be moved. | <input type="checkbox"/> |

Usage example

In this example, the API reorders the row for the referenced list page with the provided indexes and inserts the MOVE instruction for the submit action.

```
getPConnect().getListActions().reorder(1, 0);
```

The reordered `EmployeeDetails` list in Redux is as shown below:

```
EmployeeDetails: [{
  classID: 'ON8TTL-C11nGall-Data...ableData',
  Name: 'Ben',
  Experience: 2
},
{
  classID: 'ON8TTL-C11nGall-Data...ableData',
  Name: 'John',
  Experience: 2
}]
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [{
  target: ".EmployeeDetails",
  content: {},
  listIndex: 1,
  instruction: "INSERT"
},
{
  target: ".EmployeeDetails",
  content: {},
```

```

    listIndex: 2,
    instruction: "INSERT"
  },
  {
    target: ".EmployeeDetails",
    content: {
      Name: 'John'
    },
    listIndex: 1,
    instruction: "UPDATE"
  },
  {
    target: ".EmployeeDetails",
    content: {
      Name: 'Ben'
    },
    listIndex: 2,
    instruction: "UPDATE"
  },
  {
    target: ".EmployeeDetails",
    listIndex: 2,
    listMoveToIndex: 1
    instruction: "MOVE"
  },
]

```

replacePage(property, payload)

Adds the REPLACE page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|-----------------|--------|---|--------------------------|
| property | string | The name of the referenced list to which the REPLACE page instruction is added. | <input type="checkbox"/> |
| payload | object | The object containing the property and its value to be replaced in the referenced list. | <input type="checkbox"/> |

Usage example

In this example, the API changes the data present in the `pyCountry` property and resets the associated fields in the `ShippingAddress` page.

```
getPConnect().getListActions().replacePage('.ShippingAddress', {pyCountry : IND"})
```

setSelectedRows(rows, options)

Updates the Redux Store on selection of rows and adds the SELECT/DESELECT page instruction to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|----------------|--------|--|--------------------------|
| rows | array | The array containing the details of the selected/deselected rows. | <input type="checkbox"/> |
| options | object | The object containing additional information to perform the SELECT/DESELECT operation. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|-----------------------------|---------|---|--------------------------|
| skipPageInstructions | boolean | <p>The flag that determines if the page instructions must be added to the referenced list.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipPageInstructions is <code>true</code>, the page instructions are not added. • If skipPageInstructions is <code>false</code>, the page instructions are added. </div> | <input type="checkbox"/> |

Usage example

In this example, the API selects the rows for the given payload for the referenced list page and inserts the INSERT/DELETE instruction for the submit action.

```
getPConnect().getListActions().setSelectedRows(
  [
    {
      $key: '0759409f-4146-439c-aa25-57d4f495fee5',
      $selected: true,
      Name: 'Alice'
      pyGUID: '0759409f-4146-439c-aa25-57d4f495fee5'
    }
  ]
);
```

The selected row is inserted in the `Employees` list in Redux as shown below:

```
Employees: [{
  pyGUID: '0759409f-4146-439c-aa25-57d4f495fee5'
}]
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [{
  "target": ".Employees",
  "content": {
    pyGUID: '0759409f-4146-439c-aa25-57d4f495fee5'
  },
  "listIndex": 1,
  "instruction": "INSERT"
}]
```

setVisibility(isVisible)

Communicates if the referenced editable list is mounted on the screen to facilitate generation of page instructions.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|------------------|---------|---|--------------------------|
| isVisible | boolean | <p>The flag that determines if the referenced list is mounted on the screen.</p> <ul style="list-style-type: none"> • If isVisible is set to <code>true</code>, the referenced editable list is mounted on the screen, allowing page instructions to be generated. • If isVisible is set to <code>false</code>, the referenced editable list is not mounted on the screen due to which page instructions cannot be generated. | <input type="checkbox"/> |

Usage example

In this example, the API communicates that the referenced editable list is mounted on the screen, allowing the page instructions to be generated.

```
getPConnect().getListActions().setVisibility(true)
```

update(payload, index, options)

Adds the UPDATE page instruction to the referenced list in context.

Returns

Not applicable.



Parameters

| Name | Type | Description | Required |
|----------------|--------|---|--------------------------|
| payload | object | The object containing the values to be updated in the referenced list. | <input type="checkbox"/> |
| index | number | The position in the referenced list that must be updated. | <input type="checkbox"/> |
| options | object | The object containing additional information to perform the UPDATE operation. | <input type="checkbox"/> |

The following table contains the properties of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|--|--------------------------|
| skipStateUpdate | boolean | <p>The flag that determines if the Redux State must be updated.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> • The default value is <code>false</code>. • If skipStateUpdate is <code>true</code>, the Redux State is not updated. • If skipStateUpdate is <code>false</code>, the Redux State is updated. </div> | <input type="checkbox"/> |
| pageRef | string | The reference to the page within the current context. | <input type="checkbox"/> |

Usage example

In this example, the API updates a row for list page with the provided payload and inserts the UPDATE instruction for the submit action.

```
getPConnect().getListActions().update({
  Name: 'John',
  Experience: 2
}, 0, {
  skipStateUpdate: false,
  pageRef: "caseInfo.content.Employee"
});
```

The updated `EmployeeDetails` list in Redux is as shown below:

```
EmployeeDetails: [{
  classID: 'ON8TTL-C11nGall-Data...ableData',
  Name: 'John',
  Experience: 2
}]
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [
  {
    target: ".EmployeeDetails",
    content: {
      Name: 'John',
      Experience: 2
    },
    listIndex: 1,
    instruction: "UPDATE"
```

```
}
]
```

updateProperty(propertyName, value, options)

Adds the UPDATE page instruction for a specific field in a row to the referenced list in context.

Returns

Not applicable.

Parameters

| Name | Type | Description | Required |
|---------------------|--------|---|--------------------------|
| propertyName | string | The property name of the field that must be updated. | <input type="checkbox"/> |
| value | string | The value of the field that must be updated. | <input type="checkbox"/> |
| options | object | The object containing additional information to perform the UPDATE operation. | <input type="checkbox"/> |

The following table contains the property of the **options** object:

| Name | Type | Description | Required |
|------------------------|---------|---|--------------------------|
| skipStateUpdate | boolean | <p>The flag that determines if the Redux State must be updated.</p> <div> <p>NOTE:</p> <ul style="list-style-type: none"> The default value is <code>false</code>. </div> | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|---|----------|
| | | <ul style="list-style-type: none"> • If skipStateUpdate is <code>true</code>, the Redux State is not updated. • If skipStateUpdate is <code>false</code>, the Redux State is updated. | |

Usage example

In this example, the API updates the `Name` property in the row for the referenced list page with the provided payload and inserts the UPDATE instruction for the submit action.

```
getPConnect().getListActions().updateProperty('.Name', 'John Doe');
```

The updated `Name` property in the `EmployeeDetails` list in Redux is as shown below:

```
EmployeeDetails: [{
  classID: 'ON8TTL-C11nGall-Data...ableData',
  Name: 'John Doe',
  Experience: 2
}]
```

The page instructions for the submit action is as shown below:

```
pageInstructions: [
  {
```

```

    target: ".EmployeeDetails",
    content: {
      Name: 'John Doe'
    },
    listIndex: 1,
    instruction: "UPDATE"
  }
]

```

APIs in the ValidationApi class

Use the API in the ValidationApi class to handle validations on a field.

- **validate(value, propertyName)**

validate(value, propertyName)

Validates the specified field and updates the state if the validation is successful.

The validation status is passed to the component through the **validatemessage** prop parameter. If the validation is successful, an empty string is returned. If the validation fails, the **validatemessage** prop parameter contains the error message.

In addition to directly invoking the validate(value, propertyName) API, the field is validated automatically when the `onBlur` and/or `onChange` callbacks provided in the component prop parameter are invoked. These callbacks are either passed to the display component or called from the DX component.

Use the following code in the DX component for automatic validation where the display component invokes the callbacks provided in the props:

```

const {label, required, validatemessage, status, onChange, onBlur, readOnly, testId} =
  props;

<TextField

```



```

onChange={onChange}
onBlur={
  (event) =>{
    if (!value && event.target.value === ""){
      getPConnect().getValidationApi().validate("");
    }else if (value !== event.target.value && !readOnly){
      getPConnect().getValidationApi().validate(event.target.value);
    }
  }
}
label={label}

'''
/>

```

There are two types of client validations:

- **Required** - This validation prevents submitting empty fields.
- **Type** - This validates the field based on its type.

The following type validators are supported:

- alpha
- alphaNumeric
- alphaNumericWithSpace
- datetime
- email
- futureDate
- integer
- nonNegative
- nonNegativeInteger
- notFutureDate
- phone
- posDecimal

- required
- urgencyValue
- url
- validators
- validPhoneNumber

Returns

An object containing the validation status.

NOTE:

- If the field is successfully validated, the returned object will contain the validation status as true.

```
{
  status: true
}
```




- If the field fails the validation, the returned object will contain an error message and the validation status as false.

```
{
  message:"Error message description",
  status: false
}
```

Parameters

| Name | Type | Description | Required |
|--------------|--------|---|--------------------------|
| value | string | The value of the field to be validated. | <input type="checkbox"/> |
| propertyName | string | The name of the field to be validated. | <input type="checkbox"/> |

| Name | Type | Description | Required |
|------|------|---|----------|
| | | <div>  NOTE: If propertyName is not provided, the default value is obtained from the Redux state. </div> | |

Usage example

In this example, the API validates if `abcmail.com` can be entered as a value in an `email` field.

```
pConnect.getValidationApi().validate('abcmail.com', '.email');
```

The API returns the following object containing the validation status and an error message.

```
{
  message:"Invalid email address",
  status: false
}
```