Using PCore and PConnect Public APIs '24.2

5 September 2025

CONTENTS

Accessing APIs from the PCore object

The PCore APIs are global APIs that are available in Constellation architecture driven applications. These APIs provide information about the application's environment, locale, and other information that is unrelated to a specific UI component.

PCore is a singleton object that provides access to the Constellation environment and helps with accessing core functionality such as accessing Redux, data, cases, operating on cases or data, initializing localization, maintaining caches, and garbage collection.

Unlike PConnect, the PCore object is not context aware. However, some APIs may require context, in which case other utility APIs can help you operate independently.

Use the PCore object to obtain access to the publicly available capabilities of the Constellation JavaScript Engine when there is no specific context or when a PConnect object is not available.

- **Directly accessed PCore APIs**
- **APIs in the ActionsSequencer class**
- **APIs in the ActiveContext class**
- **APIs in the AnalyticsUtils class**
- **APIs in the AnnotationUtils class**
- **APIs in the AssetLoader class**
- **APIs in the AsynchronousUtils class**
- **APIs in the AttachmentUtils class**
- **APIs in the AuthUtils class**

Directly accessed PCore APIs

Access APIs directly from the PCore object.

- **checkIfSemanticURL()**
- **configureForBrowserBookmark(payload)**
- **createPConnect(config)**
- **getActionsSequencer()**
- **getAnalyticsUtils()**
- **getAnnotationUtils()**
- **getAssetLoader()**
- **getAsynchronousUtils()**
- **getAttachmentUtils()**
- **getAuthUtils()**
- **getBehaviorOverride(theOverride)**
- **getBehaviorOverrides()**
- **getCascadeManager()**
- **getCaseFollowerApi()**
- **getCaseUtils()**
- **getComponentsRegistry()**
- **getConstants()**
- **getContainerUtils()**
- **getContextTreeManager()**
- **getDataApi()**

- **getDataApiUtils()**
- **getDataPageUtils()**
- **getDataTypeUtils()**
- **getEnvironmentInfo()**
- **getErrorHandler()**
- **getEvents()**
- **getExpressionEngine()**
- **getFeedUtils()**
- **getFieldDefaultUtils()**
- **getFieldUtils()**
- **getFormUtils()**
- **getGenAIAssistantUtils()**
- **getInitialiser()**
- **getLocaleUtils()**
- **getMashupApi()**
- **getMessageManager()**
- **getMessagingServiceManager()**
- **getMetadataUtils()**
- **getNavigationUtils()**
- **getPCoreVersion()**

- **getPersonalizationUtils(listId)**
- **getPubSubUtils()**
- **getRelatedCasesApi()**
- **getRestClient()**
- **getSemanticUrlUtils()**
- **getStakeholderUtils()**
- **getStateUtils()**
- **getStore()**
- **getStoreValue(propReference, pageReference, context)**
- **getTagUtils()**
- **getUserApi()**
- **getViewResources()**
- **isDeepEqual(oldValue, newValue)**
- **isValidSemanticURL()**
- **onPCoreReady(callback)**
- **registerComponentCreator(creator)**
- **registerLocaleManager(customLocaleUtilApis)**
- **setBehaviorOverride(overrideKey, overrideValue)**
- **setBehaviorOverrides(overridesObj)**

checkIfSemanticURL()

Determines if the browser URL is semantic by comparing it with the information in the metadata of the loaded application.

Returns

The Boolean value `true` if the browser URL is semantic.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value, which denotes if the browser URL is semantic.

```
PCore.checkIfSemanticURL();
```

configureForBrowserBookmark(payload)

Enables you to directly navigate to a bookmarked page.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
payload	object	An object that specifies the container that the bookmarked page should be displayed in and the context associated with the page.	☐

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

Name	Type	Description	Required
target	string	The container that the bookmarked page should be displayed in.	☐
context	string	The context associated with the page.	☐
pageReference	string	This is a constant value – `pyPortal`.	☐

Usage example

In this example, the API navigates to the bookmarked page.

```
configureForBrowserBookmark({ target: "app/primary", context: "app/primary_0", pa
geReference:"pyPortal" }
```

createPConnect(config)

Creates a PConnect object from the input configuration of a component. The PConnect object represents a newly created component context for the given input configuration and has access to all public PConnect APIs.

> **NOTE:**
>
> Use the createPConnect(config) API if you need to explicitly create a PConnect object for a component.
>
> For example, in a table, each column has its own type and configuration. To render the cells in the table, you can use the createPConnect(config) API to create a PConnect object by passing the config object that contains the details of the cells along with the context and pageReference of the parent component.

Returns

A PConnect object created from the configuration object.

Parameters

Name	Type	Description	Required
config	object	The payload used to create a PConnect object.	☐

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

Name	Type	Description	Required
meta.type	string	The type of the component.	☐
options.context	string	The name of the context under which the component is rendered. **NOTE: context** must be passed if the component has to render on a context that is different from the parent component's context.	☐
options.pageReference	string	The data reference path of the store where the data value is stored for the current component. **NOTE: pageReference** must be passed if the component has to render on a page reference that is	☐

Name	Type	Description	Required
		different from the parent component's page reference.	

Usage example

In this example, the API creates a PConnect object for a dropdown component based on the input configuration of the config object.

```
const config = {
 "meta": {
   "type": "DropDown",
   "config": {
     label: "@L Type"
   }
  },
  "options": {
   context: "contextName",
   pageReference: "pageRef"
   }
}

const dropDownPConn = createPConnect(config);
```

getActionsSequencer()

Obtains an entry point to the ActionsSequencer object that contains APIs to sequence different types of actions in the Constellation architecture.

To view the APIs in the ActionsSequencer class, see APIs in the ActionsSequencer class.

Returns

The ActionsSequencer object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ActionsSequencer object.

```
PCore.getActionsSequencer();
```

getAnalyticsUtils()

Obtains an entry point to the AnalyticsUtils object that contains the APIs that help analytics entities or actions perform data interactions with the PRPC server.

To view the APIs in the AnalyticsUtils class, see APIs in the AnalyticsUtils class.

Returns

The AnalyticsUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AnalyticsUtils object containing the utility APIs.

```
PCore.getAnalyticsUtils();
```

getAnnotationUtils()

Obtains an entry point to the AnnotationUtils class that contains utility APIs to handle the annotation to a property.

To view the APIs in the AnnotationUtils class, see APIs in the AnnotationUtils class.

Returns

The AnnotationUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AnnotationUtils class containing the utility APIs.

```
PCore.getAnnotationUtils();
```

getAssetLoader()

Obtains an entry point to the AssetLoader object that contains the APIs to load script files and CSS files to the browser's Document Object Model (DOM).

To view the APIs in the AssetLoader class, see APIs in the AssetLoader class.

Returns

The AssetLoader object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AssetLoader object containing the utility APIs.

```
PCore.getAssetLoader();
```

getAsynchronousUtils()

Obtains an entry point to the AsynchronousUtils object that contains utility APIs that perform asynchronous operations using Observable patterns.

To view the APIs in the AsynchronousUtils class, see APIs in the AsynchronousUtils class.

Returns

The AsynchronousUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AsynchronousUtils object.

```
PCore.getAsynchronousUtils();
```

getAttachmentUtils()

Obtains an entry point to the AttachmentUtils object that contains utility APIs that handle the attachments of a case.

To view the APIs in the AttachmentUtils class, see APIs in the AttachmentUtils class.

Returns

The AttachmentUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AttachmentUtils object.

```
PCore.getAttachmentUtils();
```

getAuthUtils()

Obtains an entry point to the AuthUtils object that contains APIs to handle authentication tokens that are utilized for REST API calls.

To view the APIs in the AuthUtils class, see APIs in the AuthUtils class.

Returns

The AuthUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the AuthUtils object.

```
PCore.getAuthUtils();
```

getBehaviorOverride(theOverride)

Obtains the current value of the requested behavior override flag.

For a description of the flag that is available, see setBehaviorOverrides(overridesObj).

Returns

A Boolean value.

Parameters

Name	Type	Description	Required
theOverride	string	The override flag whose value is being requested. **NOTE:** • If the value of **theOverride** has been set with a call to the setBehaviorOverrides(overridesObj) API, the current value of the requested override is returned. • If the value of **theOverride** has not been set, this will return undefined.	☐

Usage example

If a previous call was made to

`PCore.setBehaviorOverrides( { "dynamicLoadComponents": false } );`,

then a subsequent call to

`PCore.getBehaviorOverride("dynamicLoadComponents");`

will return the Boolean value `false`.

getBehaviorOverrides()

Obtains a JSON object containing the current behavior override flags and their current values.

Returns

A JSON object.

Parameters

This API does not have parameters.

Usage example

If no previous call was made to

`PCore.setBehaviorOverrides();`,

then a call to

`PCore.getBehaviorOverrides();`

will return an empty object, because no behavior overrides were set.

If a previous call was made to

`PCore.setBehaviorOverrides( { "dynamicLoadComponents": false } );`,

then a call to

`PCore.getBehaviorOverrides();`

will return `{ "dynamicLoadComponents": false }`

getCascadeManager()

Provides an entry point to the CascadeManager object that contains the APIs to handle data page parameters for callback subscription.

To view the APIs in the CascadeManager class, see APIs in the CascadeManager class.

Returns

The CascadeManager object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the CascadeManager object.

```
PCore.getCascadeManager();
```

getCaseFollowerApi()

Provides an entry point to the CaseFollowerApi object that contains utility APIs to handle the followers of a case.

To view the APIs in the CaseFollowerApi class, see APIs in the CaseFollowerApi class.

Returns

The CaseFollowerApi object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the CaseFollowerApi object.

```
PCore.getCaseFollowerApi();
```

getCaseUtils()

Provides an entry point to the CaseUtils object that is used to access the utility APIs that perform case-related actions.

To view the APIs in the CaseUtils class, see APIs in the CaseUtils class.

Returns

The CaseUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the CaseUtils object.

```
PCore.getCaseUtils();
```

getComponentsRegistry()

Obtains the components registry that is bound to the Constellation JavaScript Engine. The components registry contains methods to register or override existing components.

Returns

The components registry as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the componentRegistry object that contains methods to register or override existing components.

```
const componentRegistry = PCore.getComponentsRegistry();
```

getConstants()

Obtains the public constants (containing un-modifiable constant objects) that can be used where specified by PCore.

To view the public constants, see List of public constants.

Returns

The constants as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the array of assignments that belongs to the current case is returned.

```
const constants = PCore.getConstants();
PConnect.getValue(constants.CASE_INFO.ASSIGNMENTS);
```

getContainerUtils()

Provides an entry point to the ContainerUtils object that is used to access the container APIs of the Constellation JavaScript Engine.

To view the APIs in the ContainerUtils class, see APIs in the ContainerUtils class.

Returns

The ContainerUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the ContainerUtils object.

```
PCore.getContainerUtils();
```

getContextTreeManager()

Obtains an entry point to the ContextTreeManager object that contains APIs to register and handle mutations confined to a view and its children.

To view the APIs in the ContextTreeManager class, see APIs in the ContextTreeManager class.

Returns

The ContextTreeManager object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the entry point to the ContextTreeManager object.

```
PCore.getContextTreeManager();
```

getDataApi()

Obtains the metadata associated with the data source contained in the config object.

> **NOTE:** This API must be used in conjunction with the init method.

Returns

The DataApi object containing the metadata and a helper method fetchData that is used to get the actual data from the data source.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the metadata associated with the data source contained in the `dataConfig` object.

```
PCore.getDataApi().init(dataConfig);
```

getDataApiUtils()

Provides an entry point to the DataApiUtils object that contains utility APIs to retrieve information from data views.

To view the APIs in the DataApiUtils class, see APIs in the DataApiUtils class.

Returns

The DataApiUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the entry point to the DataApiUtils object.

```
PCore.getDataApiUtils();
```

getDataPageUtils()

Provides an entry point to the DataPageUtils object that contains utility APIs to retrieve data from data pages.

To view the APIs in the DataPageUtils class, see APIs in the DataPageUtils class.

Returns

The DataPageUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the entry point to the DataPageUtils object.

```
PCore.getDataPageUtils();
```

getDataTypeUtils()

Provides an entry point to the DataTypeUtils object that contains utility APIs to retrieve information about data types.

To view the APIs in the DataTypeUtils class, see APIs in the DataTypeUtils class.

Returns

The DataTypeUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the entry point to the DataTypeUtils object.

```
PCore.getDataTypeUtils();
```

getEnvironmentInfo()

Obtains an entry point to the EnvironmentInfo object that contains APIs to retrieve information about the application environment that the user is currently logged into.

To view the APIs in the EnvironmentInfo class, see APIs in the EnvironmentInfo class.

Returns

The EnvironmentInfo object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the EnvironmentInfo object.

```
PCore.getEnvironmentInfo();
```

getErrorHandler()

Obtains an entry point to the ErrorHandler object that contains APIs to handle errors.

To view the APIs in the ErrorHandler class, see APIs in the ErrorHandler class.

Returns

The ErrorHandler object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the ErrorHandler object.

```
PCore.getErrorHandler();
```

getEvents()

Obtains an entry point to the Events object that contains APIs to subscribe to various events.

To view the APIs in the Events class, see APIs in the Events class.

Returns

The Events object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the Events object.

```
PCore.getEvents();
```

getExpressionEngine()

Obtains an entry point to the ExpressionEngine object that contains APIs to perform expression-related actions.

To view the APIs in the ExpressionEngine class, see APIs in the ExpressionEngine class.

Returns

The ExpressionEngine object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the ExpressionEngine object.

```
PCore.getExpressionEngine();
```

getFeedUtils()

Obtains an entry point to the FeedUtils object that contains APIs to handle the feeds of a case.

To view the APIs in the FeedUtils class, see APIs in the FeedUtils class.

Returns

The FeedUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the FeedUtils object.

```
const feedUtils = PCore.getFeedUtils();
```

getFieldDefaultUtils()

Obtains an entry point to the FieldDefaultUtils object that contains APIs to handle the operations related to the default configuration of a field type.

To view the APIs in the FieldDefaultUtils class, see APIs in the FieldDefaultUtils class.

Returns

The FieldDefaultUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the FieldDefaultUtils object.

```
const fieldDefaultUtils = PCore.getFieldDefaultUtils();
```

getFieldUtils()

Obtains an entry point to the FieldUtils object that contains APIs to handle field-related operations.

To view the APIs in the FieldUtils class, see APIs in the FieldUtils class.

Returns

The FieldUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the FieldUtils object.

```
const fieldUtils = PCore.getFieldUtils();
```

getFormUtils()

Obtains an entry point to the FormUtils object that contains APIs that handle form-related cases.

To view the APIs in the FormUtils class, see APIs in the FormUtils class.

Returns

The FormUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the FormUtils object.

```
PCore.getFormUtils();
```

getGenAIAssistantUtils()

Obtains an entry point to the GenAIAssistantUtils object that contains APIs to handle the GenAI Assistant.

To view the APIs in the GenAIAssistantUtils class, see APIs in the GenAIAssistantUtils class.

Returns

The GenAIAssistantUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the GenAIAssistantUtils object.

```
PCore.getGenAIAssistantUtils();
```

getInitialiser()

Obtains an entry point to the Initialiser object that contains APIs to initialize default configurations and containers. For more information on containers, see Working with Containers.

To view the APIs in the Initialiser class, see APIs in the Initialiser class.

Returns

The Initialiser object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the Initialiser object.

```
PCore.getInitialiser();
```

Related concepts

- Initializing the Constellation environment

getLocaleUtils()

Obtains an entry point to the LocaleUtils object that contains APIs to create, update, and lookup the localization store.

To view the APIs in the LocaleUtils class, see APIs in the LocaleUtils class.

Returns

The LocaleUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the LocaleUtils object.

```
PCore.getLocaleUtils();
```

getMashupApi()

Obtains an entry point to the MashupApi object that contains APIs that help in creating cases or working with views in other environments.

To view the APIs in the MashupApi class, see APIs in the MashupApi class.

Returns

The MashupApi object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the MashupApi object.

```
PCore.getMashupApi();
```

getMessageManager()

Obtains an entry point to the MessageManager class that contains APIs to access and manipulate messages from the Redux Store.

To view the APIs in the MessageManager class, see APIs in the MessageManager class.

Returns

The MessageManager object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the MessageManager class.

```
PCore.getMessageManager();
```

getMessagingServiceManager()

Obtains an entry point to the MessagingServiceManager object that contains APIs to interact with the Constellation Messaging Service.

To view the APIs in the MessagingServiceManager class, see APIs in the MessagingServiceManager class.

Returns

The MessagingServiceManager object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the MessagingServiceManager object.

```
PCore.getMessagingServiceManager();
```

getMetadataUtils()

Obtains an entry point to the MetadataUtils class that contains APIs to retrieve metadata from the Store.

To view the APIs in the MetadataUtils class, see APIs in the MetadataUtils class.

Returns

The MetadataUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the MetadataUtils class.

```
PCore.getMetadataUtils();
```

getNavigationUtils()

Obtains an entry point to the NavigationUtils class that contains APIs to maintain the state of UI components.

To view the APIs in the NavigationUtils class, see APIs in the NavigationUtils class.

Returns

The NavigationUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the NavigationUtils class.

```
PCore.getNavigationUtils();
```

getPCoreVersion()

Obtains the version number of the PConnect and PCore APIs for the currently loaded implementation.

Returns

The version number as a string.

Parameters

This API does not have parameters.

Usage example

In this example, if the API returns “8.6.1”, then the currently loaded implementation of both PCore and PConnect are running with version “8.6.1”.

```
PCore.getPCoreVersion();
```

getPersonalizationUtils(listId)

Provides an entry point to the PersonalizationUtils object that contains utility APIs to manage the personalization instances of a list component.

To view the APIs in the PersonalizationUtils class, see APIs in the PersonalizationUtils class.

Returns

The PersonalizationUtils object.

Parameters

Name	Type	Description	Required
listId	string	A unique ID referencing a list component. **NOTE:** The length of the ID should be limited to 32 characters.	☐

Usage example

In this example, the API obtains the PersonalizationUtils object for the list whose ID is `443533r555`.

```
const listId = "443533r555";
PCore.getPersonalizationUtils(listId);
```

getPubSubUtils()

Provides an entry point to the PubSubUtils object that contains utility APIs in the Constellation JavaScript Engine that publish and subscribe events.

To view the APIs in the PubSubUtils class, see APIs in the PubSubUtils class.

Returns

The PubSubUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the PubSubUtils object containing the utility APIs that publish and subscribe events.

```
PCore.getPubSubUtils();
```

getRelatedCasesApi()

Obtains an entry point to the RelatedCasesApi object that contains APIs to handle the related cases of a case.

To view the APIs in the RelatedCasesApi class, see APIs in the RelatedCasesApi class.

Returns

The RelatedCasesApi object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the RelatedCasesApi object containing the APIs that handle the related cases of a case.

```
PCore.getRelatedCasesApi();
```

getRestClient()

Obtains an entry point to the RestClient object that contains APIs that utilize the service broker to manage REST API calls.

To view the APIs in the RestClient class, see APIs in the RestClient class.

Returns

The RestClient object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the RestClient object containing the APIs that utilize the service broker to manage REST API calls.

```
PCore.getRestClient();
```

getSemanticUrlUtils()

Obtains an entry point to the SemanticUrlUtils object that contains APIs to build semantic URLs.

To view the APIs in the SemanticUrlUtils class, see APIs in the SemanticUrlUtils class.

Returns

The SemanticUrlUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the SemanticUrlUtils object.

```
PCore.getSemanticUrlUtils();
```

getStakeholderUtils()

Obtains an entry point to the StakeholderUtils object that contains APIs to handle the participants of a case.

To view the APIs in the StakeholderUtils class, see APIs in the StakeholderUtils class.

Returns

The StakeholderUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the StakeholderUtils object.

```
PCore.getStakeholderUtils();
```

getStateUtils()

Provides an entry point to the StateUtils object that is used to access the utility APIs that perform actions related to the state of the Store.

To view the APIs in the StateUtils class, see APIs in the StateUtils class.

Returns

The StateUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the StateUtils object.

```
PCore.getStateUtils();
```

getStore()

Obtains the Redux Store. The current data representation of the Redux store determines the rendering of the UI.

Returns

The Redux Store as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the list of functions in the Redux Store.

```
const store = pCore.getStore();
```

getStoreValue(propReference, pageReference, context)

Obtains the value of a property from the Redux Store.

Returns

The value of the property.

Parameters

Name	Type	Description	Required
propReference	string	The name of the property whose value is returned from the Redux Store.	☐
pageReference	string	The name of the page where **propReference** is located.	☐
context	string	The name of the context where **pageReference** is located.	☐

Usage example

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

```
const value = PCore.getStoreValue('.firstName', 'caseInfo.content', 'app/primary_1');
```

getTagUtils()

Obtains an entry point to the TagUtils object that contains APIs to handle the tags of a case.

To view the APIs in the TagUtils class, see APIs in the TagUtils class.

Returns

The TagUtils object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the TagUtils object.

```
const { getTaggedCases, getTags, postTags, deleteTag } = PCore.getTagUtils();
```

getUserApi()

Obtains an entry point to the UserApi object that contains APIs to handle user data.

To view the APIs in the UserApi class, see APIs in the UserApi class.

Returns

The UserApi object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the UserApi object.

```
PCore.getUserApi();
```

getViewResources()

Obtains an entry point to the ViewResources object that contains APIs to manage view metadata in the rule store.

To view the APIs in the ViewResources class, see APIs in the ViewResources class.

Returns

The ViewResources object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains an entry point to the ViewResources object.

```
PCore.getViewResources();
```

isDeepEqual(oldValue, newValue)

Determines if the values of two objects are the same by performing a deep comparison.

Returns

The Boolean value `true` if the values of the objects are the same.

Parameters

Name	Type	Description	Required
oldValue	object	The value of the first object.	☐
newValue	object	The value of the second object.	☐

Usage example

In this example, the API returns the Boolean value `true`.

```
PCore.isDeepEqual({'a': '123'}, {'a': '123'});
```

isValidSemanticURL()

Determines if a browser URL is a valid semantic URL by comparing it with the corresponding entry in the routing table.

Returns

The Boolean value `true` if the browser URL is a valid semantic URL.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Boolean value that indicates if the URL is a valid semantic URL.

```
AppRouter.isValidSemanticURL();
```

onPCoreReady(callback)

Registers a callback that will be called once the application infrastructure is ready to perform its initial render.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
callback	function	Function that will be called when the application infrastructure is ready to perform its initial render. **NOTE:** The callback has two arguments – root and target. • The root is a JSON object containing a `props` key. This is the component object that provides access to the initial getPConnect API for the top-level component of the application. The root may also contain a `domContainerID` key, which is used to get the DOM element with the given ID which is then used as the target element in the order that the component object will be rendered. If the key is null, the target argument passed to the callback is used as the initial point of rendering. • The target is null when the application is first being rendered and may be a string such `app/primary` to	☐

Name	Type	Description	Required
		indicate the target container in which the application should be rendered. If target is not defined, the component object is rendered in the DOM's element or if there is no element, it is rendered in the element with the `app` ID.	

Usage example

In this example, the API passes the incoming root and target arguments to the method that renders the application.

```
PCore.onPCoreReady((root, target) => {render(root, target)});
```

registerComponentCreator(creator)

Registers the component creator function.

> **NOTE:** The registered component creator function receives the `C11nEnv` object as the primary argument and the `additionalProps` object (if any) as the secondary argument.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
creator	function	The function that is called to register a component creator. This takes the resolved properties from the createComponent(componentMeta, dataSource, index, additionalPropsToComp) API and renders the component.	☐

Usage example

In this example, the API registers the component creator function, which will be used by all rendered components to create the component instance.

```
PCore.registerComponentCreator((c11nEnv, additionalProps = {}) => {
    return React.createElement(PConnectHOC(), {
      ...c11nEnv,
      ...c11nEnv.getPConnect().getConfigProps(),
      ...c11nEnv.getPConnect().getActions(),
      ...{
        additionalProps
      }
    });
```

registerLocaleManager(customLocaleUtilApis)

Registers the custom APIs that override the out of the box LocaleUtils APIs.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
customLocaleUtilApis	object	The APIs that will override the existing LocaleUtils APIs.	☐

Usage example

In this example, the API registers the custom APIs that override the existing LocaleUtils APIs.

```
PCore.registerLocaleManager(customLocaleUtilApis);

customLocaleUtilApis = {
  "setLocaleForRule" : function(){ //custom implementation},
  "getLocaleForRule" : function(){ //custom implementation},
  "resetLocaleStore" : function(){ //custom implementation},
  "getLocaleValue"   : function(){ //custom implementation},
  "setTimezone"      : function(){ //custom implementation},
  "getTimeZoneInUse" : function(){ //custom implementation}
}
```

setBehaviorOverride(overrideKey, overrideValue)

Sets or updates flags to override default behavior at runtime.

NOTE:

- This API adds or updates the value of the provided override key. The values of the other existing override keys are not updated. For example, if the `dynamicLoadComponents` override has been set to `false` and

you call `setBehaviorOverride( “dynamicSetCookie”, false)`, the value of `dynamicLoadComponents` is left unchanged.

- Currently, only the following override flags are supported:
  - The `dynamicLoadComponents` override flag has a default value of `true`. When set to true, the runtime will load all components that have not already been loaded when the content on the `page` in the single page application changes and indicates that one or more component types are required for that page. When set to `false`, the runtime will not attempt to load components dynamically, and assumes that all components will be loaded as part of the application’s initial load.
  - The `dynamicSemanticUrl` override flag has a default value of `true`. When set to `true`, a semantic URL for the single page application `page` is generated when needed. When set to `false`, the runtime will not attempt to update the application’s semantic URL dynamically. This can be useful when it would be inappropriate to have a change in the mashup page affect the URL of outer page that contains the mashup.
  - The `dynamicSetCookie` override flag has a default value of `true`. When set to `true`, the application establishes a cookie for the application that assists with access to the static content server. When set to `false`, the runtime will not attempt to set the application’s C11n cookie dynamically. This can be useful when we want to use the cookies of the application that contains the mashup.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
overrideKey	string	The key value for the behavior override flag.	☐
overrideValue	Boolean	The desired value of the override key.	☐

Usage example

In this example, the API sets the `dynamicLoadComponents` override to `false`. While this override is set to `false`, the application will not try to dynamically load any components that are marked as required for the single page application's `page`. It is assumed that all components that may be encountered have already been loaded.

```
PCore.setBehaviorOverride("dynamicLoadComponents", false);
```

setBehaviorOverrides(overridesObj)

Sets flags to override default behavior at runtime.

NOTE:

- This API replaces all existing values of the override keys with the newly specified override keys/values. For example, if overrides had previously been set for `dynamicLoadComponents` and the `dynamicSetCookie` (each set to `false`) and a call to `setBehaviorOverrides( { "dynamicSemanticUrl": false })`, the previously set overrides for `dynamicLoadComponents` and `dynamicSetCookie` would be reset back to their default values.
- The currently supported set of behavior overrides is defined in the setBehaviorOverride(overrideKey, overrideValue) API section.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
overridesObj	object	The JSON object containing the behavior override flags.	☐

Usage example

In this example, the API turns off the behavior to load components dynamically.

```
PCore.setBehaviorOverrides( { "dynamicLoadComponents": false } );
```

APIs in the ActionsSequencer class

Use the APIs in the ActionsSequencer class to sequence different types of actions in the Constellation architecture.

There are two types of UI actions in Constellation:

- **Blocking actions** - These actions can be executed independent of other UI actions that are currently being executed.
- **Deferred actions** - These actions can be executed only when there are no other blocking actions that are currently being executed.

Example

Consider an example of a medical insurance claim where you want to update the details of a patient, upload the required files, and submit the claim. After the claim is submitted, the form should display the entered details along with the uploaded files.

Here, uploading the required files is a blocking action, while submitting the claim is a deferred action.

Follow this sequence to implement the code for uploading the required files:

1. Use the registerBlockingAction API to register uploading the required files as a blocking action on a context.
2. Upload the required files.
3. Use the deRegisterBlockingAction API to de-register the above blocking action.

Follow this sequence to implement the code for submitting the claim:

1. Use the executeOrQueueDeferredAction API to submit the claim when no blocking actions are being currently executed on the defined context.
2. After the claim is submitted, use the handleDeferredActionCompletion API to signal the completion of the claim submission.

- **cancelDeferredActionsOnError(context)**
- **deRegisterBlockingAction(context)**
- **executeOrQueueDeferredAction(actionPayload)**
- **handleDeferredActionCompletion(context)**
- **registerBlockingAction(context)**

cancelDeferredActionsOnError(context)

Cancels queued deferred actions when an error occurs.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API cancels queued deferred actions when an error occurs.

```
PCore.getActionsSequencer().cancelDeferredActionsOnError("app/primary_1");
```

deRegisterBlockingAction(context)

De-registers an ongoing blocking action in the context of a container when the blocking action is completed.

Returns

A promise containing the de-registration status of the ongoing blocking action.

Parameters

Name	Type	Description	Required
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API tries to de-register the ongoing blocking action in the context of the `app/primary_1` container and returns a promise containing the de-registration status.

```
PCore.getActionsSequencer().deRegisterBlockingAction("app/primary_1").then(successCallback).catch(failureCallback);
```

executeOrQueueDeferredAction(actionPayload)

Executes or queues deferred actions in the context of a container.

Returns

A promise associated with the action.

NOTE:

- The promise resolves successfully when the deferred action has been either executed or queued.
- The promise rejects with an error message when the deferred action has been neither executed nor queued.

Parameters

Name	Type	Description	Required
actionPayload	object	The deferred action that must be executed or queued.	☐

Usage example

In this example, the API tries to execute or queue the specified deferred action in the context of the `app/primary_1` container and returns a promise containing the status of the deferred action.

```
PCore.getActionsSequencer().executeOrQueueDeferredAction({
    type: theType,
    payload: {
      context:app/primary_1
      //include other payload properties
    }
  }).then(successCallback).catch(failureCallback);
```

handleDeferredActionCompletion(context)

Signals the completion of execution of the ongoing deferred action and schedules the next deferred action in queue for execution.

Returns

A promise associated with the action.

NOTE:

- The promise resolves successfully when the execution of the deferred action has been completed.
- The promise rejects with an error message when the execution of the deferred action has not been completed.

Parameters

Name	Type	Description	Required
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API tries to signal the completion of execution of the ongoing deferred action and returns a promise associated with the action.

```
PCore.getActionsSequencer().handleDeferredActionCompletion("app/primary_1").then(successCallback).catch(failureCallback);
```

registerBlockingAction(context)

Registers an ongoing blocking action in the context of a container.

Returns

A promise containing the registration status of the ongoing blocking action.

Parameters

Name	Type	Description	Required
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API tries to register the ongoing blocking action in the context of the `app/primary_1` container and returns a promise containing the registration status.

```
PCore.getActionsSequencer().registerBlockingAction("app/primary_1").then(successCallback).catch(failureCallback);
```

APIs in the ActiveContext class

Use the APIs in the ActiveContext object to act on the active context for the currently opened container.

- **getCoreheaders(headerName)**

getCoreheaders(headerName)

Obtains the information of the specified header from the active context.

Returns

The information of the specified header as an object.

> **NOTE:** If the specified header does not exist or does not contain any information, an undefined value will be returned.

Parameters

Name	Type	Description	Required
headerName	string	The name of the header whose information needs to be retrieved.	☐

Usage example

In this example, the API returns an object containing the information in the `debugInfo` header.

```
PCore.getContainerUtils().getActiveContext().getCoreheaders('debugInfo');
```

APIs in the AnalyticsUtils class

Use the APIs in the AnalyticsUtils class to help analytics entities or actions perform data interactions with the PRPC server.

- **getDataObjects()**
- **getDataPageObjects(useCache)**
- **getDataViewMetadata(dataViewName, skipStoreCheck, associationFilter)**
- **getDefaultColumns(payload)**
- **getFieldsForDataSource(dataViewName, skipStoreCheck, contextName)**
- **getInsightByID(insightID)**
- **getInsightIDs()**
- **getPrimaryFieldsForDataSource(dataViewName, dataViewClassName)**
- **translateStrings(stringsToTranslate)**

getDataObjects()

Retrieves the list of data objects in the current application.

Returns

A Promise that resolves to a response containing the list of data objects corresponding to each case type in the current application.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Promise that resolves to a response containing the list of data objects.

```
PCore.getAnalyticsUtils().getDataObjects().then(response => {
  console.log(response.data);
}).catch(() => {
  ...
});
// console output:[{
//   classID: "OPB1HW-MyApp-Work-MyCase",
//   defaultListViewData: "D_MyCaseList",
//   (other properties like: description, name, links)
// }]
```

getDataPageObjects(useCache)

Retrieves the list of data objects in the current application.

Returns

A Promise that resolves to a response containing data objects corresponding to data pages.

Parameters

Name	Type	Description	Required
useCache	boolean	The flag that determines if the session cache must be used. **NOTE:**	☐

Name	Type	Description	Required
		• The default value is `true`. • Set **useCache** to `true` if you want to use session cache.	

Usage example

In this example, the API returns the list of classes with data pages of type list that have no parameters and which can be queried and backed by a report definition.

```
const dpObjs = PCore.getAnalyticsUtils()
  .getDataPageObjects(false)
  .then(({
    data
  }) => {
    const {
      dataPages
    } = data;
    return dataPages;
  });
```

getDataViewMetadata(dataViewName, skipStoreCheck, associationFilter)

Obtains the metadata of the given data view and caches the response in session storage.

The metadata contains high level information about the data view, such as the class it applies to, and returns the metadata of the fields from the class and the other classes with which it has associations. For more information on associations, see Associations.

Returns

The metadata as an object.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view whose metadata must be obtained.	☐
skipStoreCheck	boolean	The flag that determines if the metadata of the given data view must be fetched from the rule store or from the metadata API. **NOTE:** • The default value is `false`. • Set **skipStoreCheck** to `true` if the metadata of the given data view must be fetched from the metadata API. • Set **skipStoreCheck** to `false` if the metadata of the given data view must be fetched from the rule store. If the metadata is not found in the rule store, it is fetched from the metadata API.	☐
associationFilter	array.<string>	The list of simple or complex associations from which the fields must be fetched.	☐

Usage example

In this example, the API returns the metadata of the data view whose name is `D_BugList`.

```
const dataViewName = "D_BugList"
PCore.getAnalyticsUtils().getDataViewMetadata(dataViewName);

//The response of this API is as shown below.
 {
  "data" : {
   "classID": "PegaProjMgmt-Work-Bug",
   "className": "Bug",
   "structure": "List",
   "isQueryable": true,
   "fields": [
      {
      "description": "operator who manager assigns work to",
      "fieldID": "pyAssignedOperator",
      "fieldType": "Identifier",
      "isReadOnly": false,
      "name": "Assigned To",
      "dataType": "Identifier"
      },
      {
      "description": "This property is used to identify the work object's parent
      Backlog and should include the pyID of that work object. In future,
      it will be derived from the user's data input into UserStoryIDEntry or
      based upon the context of the creation of the item.",
      "displayAs": "pxTextInput",
      "fieldID": "BacklogID",
      "fieldType": "Text (single line)",
```

```
      "isReadOnly": false,
      "name": "Backlog ID",
      "maxLength": 32,
      "dataType": "Text"
      }
    ]
  }
}
```

getDefaultColumns(payload)

Retrieves the default columns for a table displaying case data.

Returns

A Promise that resolves to a response containing information for the default columns.

Parameters

Name	Type	Description	Required
payload	object	An object that specifies the report to retrieve the columns to be used as the default fields of a data entity.	☐

The following table contains the properties of the payload object:

Name	Type	Description	Required
reportName	string	The name of the report in which the default columns are configured.	☐
className	string	The class of the report in which the default columns are configured.	☐

Usage example

In this example, the API returns a Promise that resolves to a response containing information for the default columns.

```
const payload = {
  className: "OZ1CUU-MyApp-Work-MyCase",
  reportName: "DataTableEditorReport"
};
PCore.getAnalyticsUtils().getDefaultColumns(payload).then(response => {
  console.log(response.data);
}).catch(() => {
  ...
});
// console output:[
//   { pyFieldName: "pzInsKey" },
//   { pyFieldName: "pyID" },
//   { pyFieldName: "columnName" }
// ]
```

getFieldsForDataSource(dataViewName, skipStoreCheck, contextName)

Obtains the columns configured on the Report Definition bound to the given data view.

Returns

The list of columns as an object.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view through which the columns configured on the bound Report Definition are obtained.	☐
skipStoreCheck	boolean	The flag that determines if the metadata of the given data view must be fetched from the rule store or from the browser cache. **NOTE:** • Set **skipStoreCheck** to `true` if the metadata of the given data view must be fetched from the browser cache. • Set **skipStoreCheck** to `false` if the metadata of the given data view must be fetched from the rule store. If the metadata is not found in the rule store, it is fetched from the browser cache.	☐
contextName	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns a Promise that resolves to a response containing the list of columns configured on the Report Definition bound to the `D_MyCaseList` data view.

```
const dataViewName = "D_MyCaseList"
PCore.getAnalyticsUtils().getFieldsForDataSource(dataViewName, false, 'app/Primary
_1');
//The response of this API is as shown below:
 {
  "data": [
   { pyFieldName: "pzInsKey" },
   { pyFieldName: "pyID" },
   { pyFieldName: "columnName" }
  ]
 }
```

getInsightByID(insightID)

Retrieves the metadata of a specific insight.

Returns

A Promise that resolves to a response containing the metadata for the specified insight.

Parameters

Name	Type	Description	Required
insightID	string	The unique ID of the insight whose metadata is being retrieved.	☐

Usage example

In this example, the API returns a Promise that resolves to a response containing the metadata for the specified insight.

```
const insightID = "124e9385-a623-4c55-ba8e-5af8cbd0ae64";
PCore.getAnalyticsUtils().getInsightByID(insightID).then(response => {
  console.log(response.data);
}).catch(() => {
  ...
});
// console output: {
//   pyInsights: [{
//     pyContent: "[stringified insight metadata json]",
//     pyName: "Insight Name",
//     pyPermissions: [{
//       pyAccessCategory: "Rule-Access-Role-Name",
//       pyAccessType: "view",
//       pyAccessValue: "TestApp:Users"
//     }]
//   }]
// }
```

getInsightIDs()

Retrieves all available insights.

Returns

A Promise that resolves to a response containing a list of insights with the related information.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a Promise that resolves to a response containing a list of insights with the related information.

```
PCore.getAnalyticsUtils().getInsightIDs().then(response => {
  console.log(response.data);
}).catch(() => {
  ...
});

// console output: {
//   pyInsights: [{
//     pyID: "124e9385-a623-4c55-ba8e-5af8cbd0ae64",
//     pyName: "Saved Insight"
//     pyCreateDateTime: "20200630T183653.784 GMT",
//     pyCreateOperator: "user@pega.com",
//     pyUpdateDateTime: "20200630T183656.330 GMT",
//     pyUpdateOperator: "user@pega.com",
//     pxObjClass: "PegaBI-API-Insight"
//   }]
// }
```

getPrimaryFieldsForDataSource(dataViewName, dataViewClassName)

Retrieves the primary fields for a data view.

> **NOTE:** When primary fields are not found for a data view, this API retrieves the list of fields associated with the data view.

Returns

An object containing the list of primary fields.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view whose primary fields must be retrieved.	☐
dataViewClassName	string	The name of the class that the data view belongs to.	☐

Usage example

In this example, the API returns an object containing the primary fields for the `D_PrimaryFieldCaseList` data view.

```
const dataViewName = "D_PrimaryFieldCaseList"
PCore.getAnalyticsUtils().getPrimaryFieldsForDataSource(dataViewName);

//The response of this API is as seen below.
 {
  "data": [
   { pxObjClass: 'Embed-ReportUIFields', pyFieldName: "pyLabel" },
   { pxObjClass: 'Embed-ReportUIFields', pyFieldName: "pyDescription" }
  ]
 }
```

translateStrings(stringsToTranslate)

Translates a list of strings.

Returns

A Promise that resolves to a response containing a grouping of key/value pairs.

Parameters

Name	Type	Description	Required
stringsToTranslate	array	The list of strings to be translated.	☐

Usage example

In this example, the API returns a Promise that resolves to a response containing a grouping of key/value pairs.

```
const stringsToTranslate = ["Hello", "Thank you"];
PCore.getAnalyticsUtils().translateStrings(stringsToTranslate).then(response => {
  console.log(response.data);
}).catch(() => {
  ...
});
// console output: { Hello: "Hola", "Thank you": "Gracias" }
```

APIs in the AnnotationUtils class

Use the APIs in the AnnotationUtils class to handle the annotation for a property.

- **getPropertyName(value)**
- **isProperty(value)**

getPropertyName(value)

Obtains the name of an annotated property.

Returns

The name of the annotated property as a string.

Parameters

Name	Type	Description	Required
value	string	An annotated property.	☐

Usage example

In this example, the API returns `pyName` as the name of the property.

```
PCore.getAnnotationUtils().getPropertyName('@P .pyName');
```

isProperty(value)

Determines if the specified value is a property.

Returns

The Boolean value `true` if the specified value is a property.

Parameters

Name	Type	Description	Required
value	string	An annotated property.	☐

Usage example

In this example, the API returns the Boolean value `true` since `pyName` is a property.

```
PCore.getAnnotationUtils().isProperty('@P .pyName');
```

APIs in the AssetLoader class

Use the APIs in the AssetLoader class to load script files and CSS files to the browser's Document Object Model (DOM).

- **getAppAlias()**
- **getConstellationServiceUrl()**
- **getLoader(name)**
- **getStaticServerUrl()**
- **getSvcComponent(name)**
- **getSvcComponentsConfig(criteriaToMatch, options)**
- **getSvcImage(key)**
- **getSvcImageUrl(key)**
- **getSvcLocale(locale, key)**
- **initServer(url, appUrl, b2sJWT)**
- **loadAssets(assets)**
- **loadSvcComponent(name)**
- **register(name, loaderFn)**
- **setAppAlias(appAlias)**

getAppAlias()

Obtains the alias of the application for the current request.

Returns

The application alias as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the alias of the application as a string for the current request.

```
AssetLoader.getAppAlias();
```

getConstellationServiceUrl()

Obtains the top-level URL of the Constellation Service.

Returns

The Constellation Service URL as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the URL of the Constellation Service that the application will access.

```
const constellationServiceUrl = PCore.getAssetLoader().getConstellationServiceUrl();
```

getLoader(name)

Obtains the loader function associated with the name of the specified custom loader.

Returns

If the name of the custom loader is not specified, the default loader function is returned.

Parameters

Name	Type	Description	Required
name	string	The name of the custom loader.	☐

Usage example

In this example, the API returns the `font-loader` function.

```
const loader = AssetLoader.getLoader('font-loader');
```

getStaticServerUrl()

Obtains the static server URL that is used to retrieve static content.

Returns

The static server URL as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the URL of the static content.

```
const staticServerURL = PCore.getAssetLoader().getStaticServerUrl();
```

getSvcComponent(name)

Calls the Constellation Service to fetch a custom component. This call is authenticated by the Constellation Service, using the issued B2S JWT token.

Returns

A Promise that resolves to the corresponding JS implementation for the component.

Parameters

Name	Type	Description	Required
name	string	The name of the component whose JS implementation we want to fetch.	☐

Usage example

In this example, the API returns a Promise that resolves to the corresponding JS implementation for the `WeightInput` component.

```
const WeightInputDefinition = await AssetLoader
    .getSvcComponent('WeightInput.js')
    .then((response) => {
      return response;
     })
     .catch(() => {
```

```
  console.error('Unable to retrieve the component!');
});
```

getSvcComponentsConfig(criteriaToMatch, options)

Calls the Constellation Service to fetch the corresponding component `config.json` file that satisfies the specified properties. This call is authenticated by the Constellation Service, using the issued B2S JWT token.

Returns

A Promise that resolves to an object with key data and value of the array of config objects matching the criteria.

Parameters

Name	Type	Description	Required
criteriaToMatch	array	The list of name-value pairs specifying the search criteria.	☐
options	object	The object that contains information to obtain additional `config-ext.json` values.	☐

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

Name	Type	Description	Required
alternateDesignSystemURL	string	The URL for an alternative design system, used for fetching additional `config-ext.json` values.	☐

Usage example

In this example, the API retrieves an operator image and creates a DOMString containing a URL representing the image Blob object.

```
const templateDefinitionConfigJson = await AssetLoader
    .getSvcComponentsConfig([{ field: 'name', value: 'OneColumn' }])
    .then((response) => {
      if (response?.data?.components?.[0]) {
        return response.data.components[0];
      }
      throw new Error('templateDefinition for OneColumn not found!');
    })
    .catch(() => {
      throw new Error('templateDefinition for OneColumn not found!');
    });
```

getSvcImage(key)

Calls the Constellation Service to fetch an image. This call is authenticated by the Constellation Service, using the issued B2S JWT token.

Returns

A Promise that resolves to the image (blob) specified as the key.

Parameters

Name	Type	Description	Required
key	string	The unique identifier of the image.	☐

Usage example

In this example, the API returns a Promise that resolves to the image (blob) specified as the key.

```
const operatorImg = await AssetLoader
    .getSvcImage('DATA-CONTENT-IMAGE USER@CONSTELLATION.COM!PNG!/OPERAT
ORIMAGES/')
    .then((data) => {
      const url = window.URL.createObjectURL(data);
      return url;
    })
    .catch(() => {
      console.error('Unable to load the image!');
    });
```

getSvcImageUrl(key)

Calls the Constellation Service to fetch an image as a blob URL. This call is authenticated by the Constellation Service, using the issued B2S JWT token.

Returns

A Promise that resolves to the image (blob URL) specified as the key.

Parameters

Name	Type	Description	Required
key	string	The unique identifier of the image.	☐

Usage example

In this example, the API returns a Promise that resolves to the image (blob URL) specified as the key.

```
const operatorImg = await AssetLoader
    .getSvcImageUrl('DATA-CONTENT-IMAGE USER@CONSTELLATION.COM!PNG!/OPE
RATORIMAGES/')
    .then((imageUrl) => {
      img.src = imageUrl;
    })
    .catch(() => {
      console.error('Unable to load the image!');
    });
```

getSvcLocale(locale, key)

Obtains the localization instance from the Constellation service for a given user locale and an instance key. This call is authenticated by the Constellation service, using the issued B2S JWT token.

Returns

A promise that resolves to a locale JSON.

Parameters

Name	Type	Description	Required
locale	string	The name of the locale for which the localization instance must be obtained.	☐
key	string	The name of the localization instance that must be obtained.	☐

Usage example

In this example, the API obtains the locale JSON for the `en_US` locale and for the `PyCaseSummary` view in the Work- class. This API call returns a promise which when resolved gives the locale JSON.

```
const localeJson = await AssetLoader.getSvcLocale('en_US', 'WORK-!VIEW!PYCASESUMMARY.json')
    .then((data) => {
      // data is the response from the constellation service which is the locale JSON
    });
```

initServer(url, appUrl, b2sJWT)

Specifies the URLs to be used to fetch UI static content.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
url	string	The URL that fetches Pega static content sourced through CDN.	☐
appUrl	string	The URL that fetches customer static content sourced through Constellation Static Content Service.	☐
b2sJWT	string	The authorization header token used to access protected resources.	☐

Usage example

In this example, the API initiates the server with the specified URL.

```
PCore.getAssetLoader().initServer('https://mirror.pegacloud.net/prweb');
```

loadAssets(assets)

Loads asset dependencies as individual promises.

Returns

A Promise for the resolved assets.

Parameters

Name	Type	Description	Required
assets	*	The list of the names of the component assets.	☐

Usage example

In this example, the API loads the `react.prod` file into the Document Object Model (DOM).

```
PCore.getAssetLoader().loadAssets('https://mirror.pegacloud.net/assets/react.prod.js')
.then(*** success ***)
.catch(*** failure ***)
```

loadSvcComponent(name)

Calls the Constellation Service to fetch a custom JavaScript component and load it into a document. This call is authenticated by the Constellation Service, using the issued B2S JWT token.

Returns

A Promise that resolves to the name of the component.

Parameters

Name	Type	Description	Required
name	string	The name of the component whose JavaScript implementation must be fetched.	☐

Usage example

In this example, the API retrieves the JavaScript implementation for the `WeightInput` component and loads it into the document in a `<script ↗` tag.

```
const WeightInputDefinition = await AssetLoader
    .loadSvcComponent('WeightInput')
    .then((componentName) => {
      return componentName;
     })
     .catch(() => {
      console.error('Unable to retrieve the component!');
     });
```

register(name, loaderFn)

Registers a custom loader to the AssetLoader class.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
name	string	The name of the custom loader that needs to be registered.	☐
loaderFn	function	The loader function that is utilized by the custom loader.	☐

Usage example

In this example, the API registers the `font-loader` to use the `loadWebFonts` function.

```
AssetLoader.register('font-loader', loadWebFonts);
```

setAppAlias(appAlias)

Assigns a value to the alias of the application for the current request.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
appAlias	string	The alias of the application to which a value must be assigned.	☐

Usage example

In this example, the API sets `SpaceTravel` as the value of the alias of the application.

```
PCore.getAssetLoader().setAppAlias('SpaceTravel');
```

APIs in the AsynchronousUtils class

Use the API in the AsynchronousUtils class to perform asynchronous operations using Observable patterns.

- **getDebouncedSubject(delay)**

getDebouncedSubject(delay)

Obtains a new observable that yields values based on the debounced delay interval provided.

Returns

An observable object.

Parameters

Name	Type	Description	Required
delay	integer	The debounced time interval at which the observable yields values.	☐

Usage example

In this example, the API obtains an observable with a debounced interval of 200 milliseconds.

```
const sub = PCore.getAsynchronousUtils().getDebouncedSubject(200);
const subscription = sub.subscribe((argumentFromNext) => {//code});
subscription.next(valueToSubscriber);
subscription.unsubscribe();
```

APIs in the AttachmentUtils class

Use the APIs in the AttachmentUtils class to handle the attachments of a case.

- **cancelRequest(fileID)**
- **deleteAttachment(attachmentID, context)**
- **downloadAttachment(attachmentID, context)**
- **editAttachment(attachmentID, attachmentMetaData, context)**
- **getAttachmentCategories(caseID, type, context)**
- **getCaseAttachments(caseID, context, includeThumbnail)**
- **linkAttachmentsToCase(caseID, attachments, attachmentType, context)**
- **uploadAttachment(file, onUploadProgress, errorHandler, context)**

cancelRequest(fileID)

Cancels the ongoing upload request for a file that is being uploaded to a server.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
fileID	string	The unique ID of the file that is being uploaded. **NOTE:** • The **fileID** is generated by the user and can be obtained from the file object of the file being uploaded. • For more information, see the **ID** property of the **file** object in the uploadAttachment(file, onUploadProgress, errorHandler, context) API.	☐

Usage example

In this example, the API cancels the upload request for the file with the ID `_xnn4f3ig1` that is being uploaded to the server.

```
PCore.getAttachmentUtils().cancelRequest('_xnn4f3ig1');
```

deleteAttachment(attachmentID, context)

Deletes an attachment from the case to which it is linked.

Returns

A Promise that deletes an attachment from the case to which it is linked.

Parameters

Name	Type	Description	Required
attachmentID	string	The ID of the attachment that needs to be deleted.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API deletes the attachment whose ID is `LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT`.

```
PCore.getAttachmentUtils().deleteAttachment('LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT', 'app/primary_1')
.then(() => {
  // success
}).catch(err => {
  // errors
});
```

downloadAttachment(attachmentID, context)

Downloads the binary content of an attachment.

Returns

The binary content of an attachment as a Promise.

Parameters

Name	Type	Description	Required
attachmentID	string	The ID of the attachment whose binary content needs to be downloaded.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns a Promise that obtains the binary content of the attachment whose ID is `LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT`.

```
PCore.getAttachmentUtils().downloadAttachment('LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT', 'app/primary_1')
.then(() => {
  // success
}).catch(err => {
  // errors
});
```

editAttachment(attachmentID, attachmentMetaData, context)

Edits an attachment.

Returns

A Promise that when resolved successfully edits the attachment.

Parameters

Name	Type	Description	Required
attachmentID	string	The ID of the attachment that needs to be edited.	☐
attachmentMetaData	object	The name and category of the attachment.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API edits the name and category of the attachment whose ID is `LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT`.

```
const data = {"name":"Case Study","category":"File"};
PCore.getAttachmentUtils().editAttachment('LINK-ATTACHMENT ON8TTL-C11NGALL-WORK AT-17001!20230912T065407.556 GMT', data, 'app/primary_1')
.then(() => {
// success
}).catch(err => {
// errors
});
```

getAttachmentCategories(caseID, type, context)

Obtains the attachment categories linked to a case.

Returns

A Promise that obtains the attachment categories linked to a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose attachment categories must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
type	string	The type of attachment.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API obtains the attachments categories linked to a case whose ID is `W-102`.

```
PCore.getAttachmentUtils().getAttachmentCategories('ORG-MYAPP-WORK W-102', 'file', 'app/primary_1')
.then(attachments => {
// attachments array
}).catch(err => {
// errors
});
```

getCaseAttachments(caseID, context, includeThumbnail)

Obtains the attachments linked to a case.

Returns

The attachments linked to a case as a Promise.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose attachments must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
context	string	The name of the context where the API is being called.	☐
includeThumbnail	boolean	The flag that determines if the thumbnail associated with the attachment is returned. **NOTE:** • The default value is `false`. • If **includeThumbnail** is `true`, the thumbnail associated with the attachment is returned as a property of the attachment object. • If **includeThumbnail** is `false`, the thumbnail	☐

Name	Type	Description	Required
		associated with the attachment is not returned.	

Usage example

In this example, the API obtains the attachments linked to a case whose ID is `W-102`.

```
PCore.getAttachmentUtils().getCaseAttachments('ORG-MYAPP-WORK W-102', 'app/primary_1')
.then(attachments => {
   // attachments array
}).catch(err => {
  // errors
});
```

linkAttachmentsToCase(caseID, attachments, attachmentType, context)

Links an uploaded file or an array of links to a case.

Returns

A Promise that links an uploaded attachment to a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case to which the uploaded attachment needs to be linked. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
attachments	array	A file or an array of links that needs to be linked to the case.	☐
attachmentType	string	The type of attachment that needs to be linked to the case.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API links the uploaded file whose ID is `234545` to the case whose ID is `W-102`.

```
const file = {
    "type": "File",
    "category": "Screenshot",
    "attachmentFieldName": "Screenshot",
    "fileType": "PNG",
    "name": "Screenshot for this issue",
    "ID": "234545"
  }
```

```
  PCore.getAttachmentUtils().linkAttachmentsToCase('ORG-MYAPP-WORK W-102', [file], "File", "app/primary_1")
  .then((attachments) => {
    // attachments
  }).catch(err => {
    // Error handling
  });
```

In this example, the API links multiple links to the case whose ID is `W-102`.

```
const url1 = {"type":"URL","category":"URL","url":"https://www.google.com","name":"Google"}
const url2 = {"type":"URL","category":"URL","url":"https://www.firefox.com","name":"Firefox"}

PCore.getAttachmentUtils().linkAttachmentsToCase('ORG-MYAPP-WORK W-102', [url1, url2], "File", "app/primary_1")
.then((attachments) => {
  // attachments
}).catch(err => {
  // Error handling
});
```

uploadAttachment(file, onUploadProgress, errorHandler, context)

Uploads a file to the server.

Returns

A Promise that resolves to an object containing the metadata of the uploaded file.

Parameters

Name	Type	Description	Required
file	object	The file that needs to be uploaded to the server.	☐
onUploadProgress	function	A callback function that provides the upload progress of the file.	☐
errorHandler	function	A callback function to handle exceptions.	☐
context	string	The name of the context where the API is being called.	☐

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

Name	Type	Description	Required
ID	string	The unique identifier of the file that must be uploaded. **NOTE:** The **ID** must be generated by the user.	☐
category	string	The sub-group of the file that must be uploaded.	☐

Usage example

In this example, the API uploads a file to the server.

```
const onUploadProgress = (id, progressInfo) => {
 // id is the unique ID provided in the file object
 // progressInfo is an object containing the progress of the file that is being uploaded
}

const errorHandler = (isRequestCancelled, file) => {
  return (error) => {
   /*
    isRequestCancelled - function to determine whether the upload request is cancelle
d
    file – the file object that was being uploaded
    error - an object containing data related to the error.
    Some of the sample properties of the error object in the event of a network failure
 are mentioned below:
      code: "ERR_NETWORK"
      message: "Network Error"
   */
   if (!isRequestCancelled(error)) {
   // your code to handle genuine errors
   }
  }
}

const uniqueID = "_xnn4f3ig1";   //unique file ID generated by user
const file = {
"lastModified" : "1716379450677",
"lastModifiedDate" : "Wed May 22 2024 17:34:10 GMT+0530 (India Standard Time) "
"name": "abc_resume.pdf",
"size": 890428,
"type": "application/pdf",
"webkitRelativePath":"",
}
```

```
file.ID = uniqueID;
file.category = "Resume"

PCore.getAttachmentUtils().uploadAttachment(file, onUploadProgress, errorHandler,
 'app/primary_1')
.then((Response) => {
/*
Response Object structure
ID: "UGVnYVBsYXRmb3JtX19BdHRhY2htZW50XzY2NTU5ZjIzMDI0ZjY4Nj"
   ID is generated in the server when the file is uploaded successfully
category: "Resume"
   category is the sub-group of the file that has been uploaded
clientFileID: "_xnn4f3ig1"
   clientFileID is the unique identifier of the file that has been uploaded
filename: "abc_resume.pdf"
   filename is the name of the file that has been uploaded
type: "File"
   the value of type is always File
*/
}).catch(err => {
  // Error handling
});
```

APIs in the AuthUtils class

Use the APIs in the AuthUtils class to handle authentication tokens that are utilized for REST API calls.

- **getAuthInstance(config)**
- **revokeTokens()**
- **setAuthorizationHeader(value)**

- setTokens(tokenInfo)

getAuthInstance(config)

Obtains an instance of the PegaAuth class that contains helper functions for the OAUTH2.0 registration in Constellation architecture-based applications.

Returns

The instance of the PegaAuth class as an object.

Parameters

Name	Type	Description	Required
config	object	The object that contains the clientID, clientSecret, and endPoints properties that help in obtaining an instance of the PegaAuth class.	☐

Usage example

In this example, the API returns an instance of the PegaAuth class.

```
const config = {
  clientId: 'clientId',
  clientSecret: 'client_secret_value',
  endPoints: {
    token: 'token_endpoint',
    revoke: 'revoke_endpoint',
    authorize: 'authorize_endpoint'
  }
};
PCore.getAuthUtils().getAuthInstance(config);
```

revokeTokens()

Revokes the access and refresh tokens.

Returns

A Promise that is resolved when the access and refresh tokens are successfully revoked.

Parameters

This API does not have parameters.

Usage example

In this example, the API revokes the access and refresh tokens.

```
PCore.getAuthUtils().revokeTokens();
```

setAuthorizationHeader(value)

Adds the authorization header to the fetch request headers to be utilized to validate all subsequent Constellation DX API calls.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
value	string	The value of the authorization header to be utilized to validate all subsequent Constellation DX API calls.	☐

Name	Type	Description	Required
		NOTE: The value must be specified in the following format: `<authentication scheme> <authorization parameters>`	

Usage example

In this example, the API adds the `Authorization: Bearer Abcdefg==` header to the fetch request headers.

```
PCore.getAuthUtils().setAuthorizationHeader('Bearer Abcdefg==');
```

setTokens(tokenInfo)

Adds the access token to the fetch request headers.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
tokenInfo	object	The object containing the access token, refresh token, session index, duration of access token expiry, and the token type.	☐

Usage example

In this example, the API adds the access token from the `tokenObject` object to the fetch headers.

```
const tokenObject = {
    access_token: 'access_token_value',
    refresh_token: 'refresh_token_value',
    session_index: 'session_index',
    expires_in: 120,
    token_type: 'bearer'
};
PCore.getAuthUtils().setTokens(tokenObject);
```

APIs in the CascadeManager class

Use the APIs in the CascadeManager class to handle data page parameters for callback subscription.

- **deregisterResetDependencies(contextName, pageReference, target, dependentProperties, fieldType)**
- **registerFields(context, pageReference, fields, callback, subscriptionId)**
- **registerListField(context, pageReference, listField, callback, subscriptionId)**
- **registerResetDependencies(contextName, pageReference, target, dependentProperties, fieldType, mode)**
- **unRegisterFields(context, pageReference, fields, subscriptionId)**
- **unRegisterListField(context, pageReference, listField, subscriptionId)**

deregisterResetDependencies(contextName, pageReference, target, dependentProperties, fieldType)

De-registers all the registered dependencies of the target field that have undergone cascade resetting. Cascade resetting is the process of resetting the value of the target field when the value of the source field changes.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
contextName	string	The name of the context containing the field whose dependencies need to be de-registered.	☐
pageReference	string	The reference to the page that contains the target field.	☐
target	string	The field whose dependencies need to be de-registered.	☐
dependentProperties	array	The fields on which the target field is dependent.	☐
fieldType	string	The type of the target field.	☐

Usage example

In this example, the API de-registers the registered dependencies of the `backlogId` target field.

```
PCore.CascadeManager.deregisterResetDependencies('app/primary_1', 'caseInfo.content', '.backlogId', ['.productId','release'], 'Text')
```

registerFields(context, pageReference, fields, callback, subscriptionId)

Registers the specified fields to the CascadeManager class.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context containing the fields to be registered.	☐
pageReference	string	The reference to the page that contains the field to be registered.	☐
fields	array	The array of fields to be registered.	☐
callback	function	The function to be called when the registered field is updated.	☐
subscriptionId	string	The unique ID for registering the fields. The same ID should be provided for de-registering the fields.	☐

Usage example

In this example, the API registers the `firstName` and `lastName` fields to trigger callback.

```
PCore.CascadeManager.registerFields('app/primary_1', 'caseInfo.content', ['firstName','lastName'], () => { console.log("field changed")}, '001-002-003')
```

registerListField(context, pageReference, listField, callback, subscriptionId)

Registers the field of type `PageList` to the CascadeManager Class.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context containing the field to be registered.	☐
pageReference	string	The reference to the page that contains the field to be registered.	☐
listField	string	The field to be registered.	☐
callback	function	The function to be called when the registered field is updated.	☐
subscriptionId	string	The unique ID for registering the field. The same ID should be provided for de-registering the field.	☐

Usage example

In this example, the API registers the `phoneNumber` field to trigger callback.

```
PCore.CascadeManager.registerListField('app/primary_1', 'caseInfo.content', phoneNumber, () => { console.log("field changed")}, '002-002-004')
```

registerResetDependencies(contextName, pageReference, target, dependentProperties, fieldType, mode)

Registers all the dependencies of the target field to perform cascade resetting. Cascade resetting is the process of resetting the value of the target field when the value of the source field changes.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
contextName	string	The name of the context containing the field whose dependencies need to be registered.	☐
pageReference	string	The reference to the page that contains the target field.	☐
target	string	The field whose dependencies need to be registered.	☐
dependentProperties	array	The fields on which the target field is dependent.	☐
fieldType	string	The type of the target field.	☐
mode	string	The selection mode of the target field based on its **fieldType**.	☐

Name	Type	Description	Required
		NOTE: The value of **mode** should be either `singleRecord` or `multiRecord`.	

Usage example

In this example, the API registers the dependencies of the `backlogId` target field.

```
PCore.CascadeManager.registerResetDependencies('app/primary_1', 'caseInfo.content', '.backlogId', ['.productId','release'], 'Text', 'singleRecord')
```

unRegisterFields(context, pageReference, fields, subscriptionId)

De-registers fields from the CascadeManager class.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context containing the fields to be de-registered.	☐
pageReference	string	The reference to the page that contains the field to be de-registered.	☐
fields	array	The array of fields to be de-registered.	☐

Name	Type	Description	Required
subscriptionId	string	The unique ID for de-registering the fields.	☐

Usage example

In this example, the API de-registers the `firstName` and `lastName` fields.

```
PCore.CascadeManager.unregisterFields('app/primary_1', 'caseInfo.content', ['firstName','lastName'], '001-002-003')
```

unRegisterListField(context, pageReference, listField, subscriptionId)

De-registers the field of type `PageList` from the CascadeManager class.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context containing the field to be de-registered.	☐
pageReference	string	The reference to the page that contains the field to be de-registered.	☐
listField	string	The field to be de-registered.	☐
subscriptionId	string	The unique ID for de-registering the field.	☐

Usage example

In this example, the API de-registers the `phoneNumber` field.

```
PCore.CascadeManager.unregisterListField('app/primary_1', 'caseInfo.content', phon
eNumber, '002-002-004')
```

APIs in the CaseFollowerApi class

Use the APIs in the CaseFollowerApi class to handle the followers of a case.

- **addCaseFollower(caseID, userID, context)**
- **deleteCaseFollower(caseID, followerID, context)**
- **getCaseFollowers(caseID, context)**

addCaseFollower(caseID, userID, context)

Adds specified users as followers to a case.

Returns

A Promise that adds the specified users as followers to a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case to which the users must be added as followers.	☐

Name	Type	Description	Required
		NOTE: Ensure that you provide the `pzInsKey` value of the **caseID**.	
userID	string[]	The IDs of the users to be added as followers to a case. **NOTE:** Specify the **userID** as an array of strings, such as `['user-1','user-2']`.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API adds a user whose ID is `user-1` as a follower to the case whose ID is `W-102`.

```
PCore.getCaseFollowerApi().addCaseFollower('ORG-MYAPP-WORK W-102', ['user-1'] , '
app/primary_1')
.then(() => {
  // success
}).catch(err => {
   // Error handling
});
```

deleteCaseFollower(caseID, followerID, context)

Deletes the follower of a case.

Returns

A Promise that deletes the follower of a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose follower must be deleted. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
followerID	string	The ID of the follower which must be deleted for a case.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API deletes the follower whose ID is `user-1` for the case whose ID is `W-02`.

```
PCore.getCaseFollowerApi().deleteCaseFollower('ORG-MYAPP-WORK W-02', 'user-1' , '
app/primary_1')
.then(() => {
  // success
```

```
}).catch(err => {
  // errors
});
```

getCaseFollowers(caseID, context)

Obtains the followers of a case.

Returns

A Promise that obtains the followers of a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose followers must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API obtains the followers of a case whose ID is `W-102`.

```
PCore.getCaseFollowerApi().getCaseFollowers('ORG-MYAPP-WORK W-102' , 'app/primary_1')
.then(followers => {
```

```
  // array containing the list of followers for the case
}).catch(err => {
  // errors
});
```

APIs in the CaseUtils class

Use the APIs in the CaseUtils class to perform case-related actions.

- **getCaseEditLock(caseID, context)**
- **getCaseEditMetadata(caseID, context)**
- **isCaseActive(key, target)**
- **updateCaseEditFieldsData(caseID, changeSet, eTag, context)**

getCaseEditLock(caseID, context)

Locks a case so that it can be edited.

Returns

A promise that when resolved indicates that acquiring the lock of the case is successful.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case that must be locked.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API obtains a successful lock on the case so that it can be edited.

```
const caseID = 'METORG-VEHICLEMANAGER-WORK V-7222';
const context = 'app/primary_1';

PCore.getDataApiUtils().getCaseEditLock(caseID, context)
.then(response => {
  //The response of this API is as shown below:
  {
    uiResources: {
      resources: {},
      components: [],
      root: {}
    },
    data: {
      caseInfo: {
        ID: // case ID,
        content: {}
      }
    }
  }
  //Having the above structure in the response indicates that the lock has been acqui
red successfully.
})
.catch(error => {
  console.log(error);
});
```

getCaseEditMetadata(caseID, context)

Obtains the edit metadata of a case.

Returns

A promise that when resolved returns the edit metadata of a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case on which the edit is being performed.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns the edit metadata of the case.

```
const caseID = "METORG-VEHICLEMANAGER-WORK V-7222";
const context = "app/primary_1";

PCore.getDataApiUtils().getCaseEditMetadata(caseID, context)
.then(response => {
  // The response of this API is as shown below:
  {
    uiResources: {},
    data: {
      caseInfo: {
          "caseTypeID": "MetOrg-VehicleManager-Work-VehiclePurchase",
          "owner": "abc@xyz.com",
          "availableActions": [],
          "lastUpdatedBy": "abc@xyz.com",
          "sla": {},
          "content": {
            "classID": "MetOrg-VehicleManager-Work-VehiclePurchase",
            "VehicleUsage": "",
            "Year": "2013",
            "VehicleType": "e7faf92e-e6b0-4793-b59f-e406a2abdb75",
```

```
            "NeededBy": "20200605T181445.999 GMT",
            "Model": "458 Italia",
            "Make": "Ferrari",
            "RequestingDepartment": "52724c8d-54b9-4819-851d-3765098adebb"
          },
          "createdBy": "abc@xyz.com",
          "createTime": "2020-05-28T20:05:41.235Z",
          "urgency": "10",
          "name": "2013 Ferrari 458 ITALIA",
          "stages": [],
          "ID": "METORG-VEHICLEMANAGER-WORK V-7222",
          "lastUpdateTime": "2020-05-28T20:05:41.541Z",
          "stageID": "PRIM3",
          "stageLabel": "Delivery",
          "status": "New"
        }
    }
  }
})
.catch(error => {
  console.log(error)
});
```

isCaseActive(key, target)

Determines if a case is active based on the information in the target container.

Returns

The Boolean value `true` if the case is active.

Parameters

Name	Type	Description	Required
key	string	The unique identifier of the case.	☐
target	string	The target container containing the details of the case.	☐

Usage example

In this example, the API returns the Boolean value `true`, if the case with the key `PEGACS-WORK-INTERACTION I-383039` is active.

```
PCore.getCaseUtils().isCaseActive('PEGACS-WORK-INTERACTION I-383039', 'app/primary');
```

updateCaseEditFieldsData(caseID, changeSet, eTag, context)

Updates the fields of a case.

> **NOTE:** To update the fields of a case, you must obtain a lock on the case using the getCaseEditLock(caseID, context) API.

Returns

A Promise that when resolved indicates that the case data is updated successfully.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose fields must be updated.	☐
changeSet	object	The object containing the data to be updated in the fields.	☐
eTag	string	The response header generated when the lock is acquired on a case successfully.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns an object indicating that the case data has been successfully updated.

```
const caseID = "METORG-VEHICLEMANAGER-WORK V-7222";
const changeSet = { "METORG-VEHICLEMANAGER-WORK V-7222": { Make: "New Value
" } };
const eTag = "20200831T114802.686 GMT";
const context = "app/primary_1";
PCore.getDataApiUtils().updateCaseEditFieldsData(caseID, changeSet, eTag, context);
.then(response => {

 // The response of this API is as shown below:
   {
    "data": {
      "caseInfo": {
        "caseTypeID": "MetOrg-VehicleManager-Work-VehiclePurchase",
        "owner": "reactuser",
```

```
        "availableActions": [],
        "lastUpdatedBy": "mohaa5",
        "assignments": [],
        "sla": {},
        "createdBy": "reactuser",
        "createTime": "2020-06-08T12:10:08.813Z",
        "urgency": "10",
        "name": "Vehicle Purchase",
        "stages": [],
        "ID": "METORG-VEHICLEMANAGER-WORK V-10001",
        "lastUpdateTime": "2020-09-01T05:52:54.225Z",
        "stageID": "PRIM5",
        "stageLabel": "Request",
        "status": "New"
      }
    },
    "confirmationNote": "Thank you! The next step in this case has been routed appr
opriately."
  }
 })
 .catch(error => {
  console.log(error);
 });
```

APIs in the ContainerUtils class

Use the APIs in the ContainerUtils class to retrieve information pertaining to a container.

For more information on containers, see Working with Containers.

- **areContainerItemsPresent(target)**

- **clearTransientData(transientItemID)**
- **closeContainerItem(containerItemID, options)**
- **getActiveContainerItemContext(target)**
- **getActiveContainerItemName(target)**
- **getActiveContext()**
- **getChildContainerItems(containerItemName)**
- **getContainerAccessOrder(target)**
- **getContainerData(target)**
- **getContainerItemName(target, key, callback)**
- **getContainerItems(target)**
- **getContainerType(context, name)**
- **getDataContextName(containerItemName)**
- **isContainerDirty(containerItemID)**
- **isContainerInitialized(context, name)**
- **isContainerItemActive(target, key, callback)**
- **isContainerItemExists(target, key, callback)**
- **purgeTransientData(transientItemID)**
- **replaceTransientData(transientObject)**
- **updateTransientData(transientObject)**

areContainerItemsPresent(target)

Determines if container items are present in a container.

Returns

The Boolean value `true` if container items are present in the container.

Parameters

Name	Type	Description	Required
target	string	The container that is searched.	☐

Usage example

In this example, the API checks if container items are present in the `app/primary` container.

```
PCore.getContainerUtils().areContainerItemsPresent('app/primary');
```

clearTransientData(transientItemID)

Deletes the data stored in a transient item.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
transientItemID	string	The ID of the transient item containing the data to be deleted.	☐

Usage example

In this example, consider the value of the data in the transient item to be `{Name: 'James Bond', department: {id: '007', type: 'agent'}}`. After the clearTransientData API is called, the data stored in the transient item is `{Name: '', department: {id: '', type: ''}}`.

```
PCore.getContainerUtils().clearTransientData('app/primary_1/workarea_1/transientItem_1');
```

closeContainerItem(containerItemID, options)

Closes a container item.

Returns

A promise containing the order in which the remaining container items were accessed.

Parameters

Name	Type	Description	Required
containerItemID	string	The container item that must be closed.	☐
options	object	The object containing additional information required to close the container item. For example, the **skipDirtyCheck** property decides if the confirm dialog box (asking if the container item can be closed without saving the changes) should be displayed. • If **skipDirtyCheck** is `true`, the confirm dialog box is not displayed.	☐

Name	Type	Description	Required
		• If **skipDirtyCheck** is `false`, the confirm dialog box is displayed if the container item is dirty.	

Usage example

In this example, the API closes the container item with the ID `app/primary_4`. The confirm dialog box is not displayed.

```
PCore.getContainerUtils().closeContainerItem("app/primary_4",{skipDirtyCheck:true});
```

getActiveContainerItemContext(target)

Obtains the context of the active container item within a specific container.

Returns

The context of the container item as a string.

Parameters

Name	Type	Description	Required
target	string	Target container containing the active container item.	☐

Usage example

In this example, the API obtains the context of the active container item under the `app/primary` container.

```
PCore.getContainerUtils().getActiveContainerItemContext('app/primary_1/workarea');
```

getActiveContainerItemName(target)

Obtains the name of the active container item within a specific container.

Returns

The name of the container item as a string.

Parameters

Name	Type	Description	Required
target	string	The target container containing the active container item.	☐

Usage example

In this example, the API obtains the name of the active container item under the `app/primary` container.

```
PCore.getContainerUtils().getActiveContainerItemName('app/primary');
```

getActiveContext()

Provides an entry point to the ActiveContext object that contains APIs that act on the active context for the currently opened container.

To view the APIs in the ActiveContext class, see APIs in the ActiveContext class.

Returns

The ActiveContext object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the entry point to the ActiveContext object.

```
PCore.getContainerUtils().getActiveContext();
```

getChildContainerItems(containerItemName)

Obtains the child container items of a specific container item.

Returns

The child container items as an array.

Parameters

Name	Type	Description	Required
containerItemName	string	The name of the container item containing child container items.	☐

Usage example

In this example, the API obtains the child container items from the `app/primary_3` container item.

```
PCore.getContainerUtils().getChildContainerItems('app/primary_3');
```

getContainerAccessOrder(target)

Obtains the access order data for a target container.

Returns

The access order data for the target container as an array.

Parameters

Name	Type	Description	Required
target	string	The target container whose access order data is required.	☐

Usage example

In this example, the API obtains the access order data for the `app/primary` container.

```
PCore.getContainerUtils().getContainerAccessOrder('app/primary');
```

getContainerData(target)

Obtains the container information for a target container.

Returns

The state object for the target container.

Parameters

Name	Type	Description	Required
target	string	The target container whose information is required.	☐

Usage example

In this example, the API obtains the container information for the `app/primary` target container.

```
PCore.getContainerUtils().getContainerData('app/primary');
```

getContainerItemName(target, key, callback)

Obtains the name of the container item if a unique key is present in the container data.

Returns

The name of the container item as a string.

Parameters

Name	Type	Description	Required
target	string	The target container whose data is searched.	☐
key	string	The unique key that is searched for in the container data.	☐
callback	ContainerCallback	The function executed for each unique key in the container item till it returns `true`, indicating that the key has been found. **NOTE:** If this parameter is not defined, a strict	☐

Name	Type	Description	Required
		comparison is used to find the key.	

Usage example

In this example, the API searches for the `R-1234` key in the `app/primary` container and obtains the resulting container item's name.

```
PCore.getContainerUtils().getContainerItemName('app/primary', 'R-1234');
```

getContainerItems(target)

Obtains information about the items in a target container.

Returns

The information about the items in a target container as an object.

Parameters

Name	Type	Description	Required
target	string	The target container containing the items whose information is required.	☐

Usage example

In this example, the API obtains information about the items in the `app/primary` target container.

```
PCore.getContainerUtils().getContainerItems('app/primary');
```

getContainerType(context, name)

Obtains the type of a container.

Returns

The type of the container as a string.

Parameters

Name	Type	Description	Required
context	string	The context of the container.	☐
name	string	The name of the container.	☐

Usage example

In this example, the API obtains the type of the container with the context `app` and name `primary`.

```
PCore.getContainerUtils().getContainerType('app', 'primary');
```

getDataContextName(containerItemName)

Obtains the name of the data context for a specific container item.

Returns

The name of the data context as a string.

Parameters

Name	Type	Description	Required
containerItemName	string	The name of the container item containing the data context.	☐

Usage example

In this example, the API obtains the name of the data context for the container item with the name `app/primary`.

```
PCore.getContainerUtils().getDataContextName('app/primary_3');
```

isContainerDirty(containerItemID)

Determines if the content within a container item has been updated.

Returns

The Boolean value `true` if the content within a container item has been updated.

Parameters

Name	Type	Description	Required
containerItemID	string	The unique identifier of the container item whose content might have been updated.	☐

Usage example

In this example, the API determines if the content in the `app/primary_1` container item has been updated.

```
PCore.getContainerUtils().isContainerDirty('app/primary_1');
```

isContainerInitialized(context, name)

Verifies if a specific container is initialized.

Returns

The Boolean value `true` if the container is initialized.

Parameters

Name	Type	Description	Required
context	string	The context of the container to be verified.	☐
name	string	The name of the container to be verified.	☐

Usage example

In this example, the API verifies if the container with the context `app` and name `primary` is initialized.

```
PCore.getContainerUtils().isContainerInitialized('app', 'primary');
```

isContainerItemActive(target, key, callback)

Verifies if a specific item is active in a container.

Returns

The Boolean value `true` if the container item is active.

Parameters

Name	Type	Description	Required
target	string	The target container to be verified.	☐
key	string	The unique key of the container item to be searched for.	☐
callback	ContainerCallback	The function executed for each unique key in the container item till it returns `true`, indicating that the key has been found. **NOTE:** If this parameter is not defined, a strict comparison is used to find the key.	☐

Usage example

In this example, the API verifies if the container item with the `R-1234` key is active in the `app/primary` container.

```
PCore.getContainerUtils().isContainerItemActive('app/primary', 'R-1234', (key, semanticURL) => { });
```

isContainerItemExists(target, key, callback)

Verifies if a specific item exists in a container.

Returns

The Boolean value `true` if the item exists in a container.

Parameters

Name	Type	Description	Required
target	string	The target container to be verified.	☐
key	string	The unique key of the container item to be searched for.	☐
callback	ContainerCallback	The function executed for each unique key in the container item till it returns `true`, indicating that the key has been found. **NOTE:** If this parameter is not defined, a strict comparison is used to find the key.	☐

Usage example

In this example, the API verifies if the container item with the `R-1234` key exists in the `app/primary` container.

```
PCore.getContainerUtils().isContainerItemExists('app/primary', 'R-1234');
```

purgeTransientData(transientItemID)

Deletes the keys and data stored in a transient item.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
transientItemID	string	The ID of the transient item containing the keys and data to be deleted.	☐

Usage example

In this example, consider the keys and data in the transient item to be `{Name: 'James Bond', department: {id: '007', type: 'agent'}}`. After the purgeTransientData API is called, the transient item becomes an empty object `{}`.

```
PCore.getContainerUtils().purgeTransientData('app/primary_1/workarea_1/transientItem_1');
```

replaceTransientData(transientObject)

Replaces the values of the specified fields and deletes the unspecified fields within a transient item.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
transientObject	object	The JSON object containing information about the transient item and the fields to be replaced.	☐

Usage example

In this example, the API replaces the values of the fields `Prop1` and `Prop2` and deletes the field `Prop3` within the `searchCriteria` transient item.

```
const containerUtils = PCore.getContainerUtils();
containerUtils.replaceTransientData({
  transientItemID: 'searchCriteria',
  data: {
    "Prop1": "valueA",
    "Prop2": "valueB"
  }
});
```

updateTransientData(transientObject)

Updates the values of the specified fields within a transient item.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
transientObject	object	The JSON object containing information about the transient item and the fields to be updated.	☐

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

Name	Type	Description	Required
transientItemID	string	The unique identifier of the transient item.	☐
data	any	The JSON object containing key value pairs which must be updated.	☐
options	object	The JSON object containing additional information for updating the fields of the transient item.	☐

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

Name	Type	Description	Required
reset	boolean	The flag that determines if the unspecified fields in the transient item must be reset. **NOTE:** • The default value is `false`. • If **reset** is `true`, the unspecified fields in the transient item will be reset.	☐

Name	Type	Description	Required
		• If **reset** is `false`, the unspecified fields in the transient item will not be reset.	

Usage example

In this example, the API updates the values of the specified fields within the `uniqueIdentifier` transient item without resetting the values of the unspecified fields.

```
const containerUtils = PCore.getContainerUtils();
containerUtils.updateTransientData({
  transientItemID: 'uniqueIdentifier',
  data: {
    "Prop1": "valueA",
    "Prop2": "valueB"
  },
  options: {
    "reset": false
  }
});
```

APIs in the ContextTreeManager class

Use the APIs in the ContextTreeManager class to register and handle mutations confined to a view and its children.

- **mutateField(context, pageName, fieldName, mutateObject)**

- **mutatePageList(context, pageName, pageListName, mutateObject)**
- **onPageListMutate(context, pageName, viewName, pageListName, callback)**
- **onViewMutate(context, pageName, viewName, callback)**

mutateField(context, pageName, fieldName, mutateObject)

Triggers the callback that has been registered using the onViewMutate(context, pageName, viewName, callback) API on any of its parent views.

> **NOTE:** Call this API when you want to propagate the changes on a field to its parent views.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context in which the field is present.	☐
pageName	string	The name of the page in which the field is present.	☐
fieldName	string	The name of the field on which the mutation has occurred.	☐
mutateObject	object	The object that is passed to the callback.	☐

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

Name	Type	Description	Required
type	string	The type of the mutation object.	☐

Usage example

In this example, the API triggers the callback set on any of its parent views.

```
PCore.getContextTreeManager().mutateField("app/primary_1/workarea_1", "caseInfo.content", ".FieldName", [{
  fieldName: "FieldName",
  type: "error",
  message: "Field can't be blank"
}]);
```

mutatePageList(context, pageName, pageListName, mutateObject)

Triggers the callback that have been registered using the onPageListMutate(context, pageName, viewName, pageListName, callback) API on any of its parent views.

> **NOTE:** Call this API when you want to propagate the changes on a pageList field to its parent views.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context in which the field is present.	☐
pageName	string	The name of the page in which the field is present.	☐
pageListName	string	The name of the page list on which the mutation has occurred.	☐
mutateObject	object	The object that is passed to the callback.	☐

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

Name	Type	Description	Required
type	string	The type of the mutation object.	☐

Usage example

In this example, the API triggers the callback set on any of its parent views.

```
PCore.getContextTreeManager().mutatePageList("app/primary_1/workarea_1", "caseInfo.content", "Page-List-Name", [{
  fieldName: "Employees",
  type: "error",
  message: "Employees can't be blank"
}]);
```

onPageListMutate(context, pageName, viewName, pageListName, callback)

Adds a callback to be executed whenever any mutation occurs to any of the page list's children.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The context name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
pageName	string	The page name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
viewName	string	The view name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
pageListName	string	The name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
callback	callback	The callback to be executed on mutation, that is when the mutatePageList(context,	☐

Name	Type	Description	Required
		pageName, pageListName, mutateObject) API is called on any field inside this page list hierarchy. Callbacks can generally be used to modify the page list's view state.	

Usage example

In this example, the API registers a callback to listen to any changes or mutations in the current page list's hierarchy.

```
PCore.getContextTreeManager().onPageListMutate("app/primary_1/workarea_1", "caseInfo.content", "View-Name", ".PageListName", (errors) => {
    setErrorStateOnPagelist((fieldErrors) => {
    const errors = [...fieldErrors];
```

onViewMutate(context, pageName, viewName, callback)

Adds a callback to be executed whenever any mutation occurs to any of the view's children.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The context name of the page list to be listened to for changes or mutations that	☐

Name	Type	Description	Required
		might occur on fields inside the current page list hierarchy.	
pageName	string	The page name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
viewName	string	The view name of the page list to be listened to for changes or mutations that might occur on fields inside the current page list hierarchy.	☐
callback	callback	The callback to be executed on mutation, that is when the mutatePageList(context, pageName, pageListName, mutateObject) API is called on any field inside this page list hierarchy. Callbacks can generally be used to modify the page list's view state.	☐

Usage example

In this example, the API registers a callback to listen to any changes or mutations in the current view's hierarchy.

```
PCore.getContextTreeManager().onViewMutate("app/primary_1/workarea_1", "caseInfo.content", "View-Name", (errors) => {
    setErrorStateOnView((fieldErrors) => {
    const errors = [...fieldErrors];
```

APIs in the DataApiUtils class

Use the APIs in the DataApiUtils class to retrieve information from data views.

- **getCaseEditLock(caseID, context)**
- **getCaseEditMetadata(caseID, context)**
- **getData(dataViewName, payload, context, options)**
- **getDataViewMetadata(dataViewName, context, associationFilter, propertyFilter)**
- **getListCount(dataViewName, payload, context)**
- **updateCaseEditFieldsData(caseID, changeSet, eTag, context)**

getCaseEditLock(caseID, context)

Locks a case so that it can be edited.

> **NOTE:** This API is deprecated. Please use the getCaseEditLock(caseID, context) API in the CaseUtils class instead.

Returns

A promise that when resolved indicates that acquiring the lock of the case is successful.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case that must be locked.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API obtains a successful lock on the case so that it can be edited.

```
const caseID = 'METORG-VEHICLEMANAGER-WORK V-7222';
const context = 'app/primary_1';

PCore.getDataApiUtils().getCaseEditLock(caseID, context)
.then(response => {
  //The response of this API is as shown below:
  {
    uiResources: {
      resources: {},
      components: [],
      root: {}
    },
    data: {
      caseInfo: {
        ID: // case ID,
        content: {}
      }
    }
  }
  //Having the above structure in the response indicates that the lock has been acqui
red successfully.
})
.catch(error => {
  console.log(error);
});
```

getCaseEditMetadata(caseID, context)

Obtains the edit metadata of a case.

> **NOTE:** This API is deprecated. Please use the getCaseEditMetadata(caseID, context) API in the CaseUtils class instead.

Returns

A promise that when resolved returns the edit metadata of a case.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case on which the edit is being performed.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns the edit metadata of the case.

```
const caseID = "METORG-VEHICLEMANAGER-WORK V-7222";
const context = "app/primary_1";

PCore.getDataApiUtils().getCaseEditMetadata(caseID, context)
.then(response => {
  // The response of this API is as shown below:
  {
    uiResources: {},
    data: {
      caseInfo: {
          "caseTypeID": "MetOrg-VehicleManager-Work-VehiclePurchase",
          "owner": "abc@xyz.com",
```

```
          "availableActions": [],
          "lastUpdatedBy": "abc@xyz.com",
          "sla": {},
          "content": {
            "classID": "MetOrg-VehicleManager-Work-VehiclePurchase",
            "VehicleUsage": "",
            "Year": "2013",
            "VehicleType": "e7faf92e-e6b0-4793-b59f-e406a2abdb75",
            "NeededBy": "20200605T181445.999 GMT",
            "Model": "458 Italia",
            "Make": "Ferrari",
            "RequestingDepartment": "52724c8d-54b9-4819-851d-3765098adebb"
          },
          "createdBy": "abc@xyz.com",
          "createTime": "2020-05-28T20:05:41.235Z",
          "urgency": "10",
          "name": "2013 Ferrari 458 ITALIA",
          "stages": [],
          "ID": "METORG-VEHICLEMANAGER-WORK V-7222",
          "lastUpdateTime": "2020-05-28T20:05:41.541Z",
          "stageID": "PRIM3",
          "stageLabel": "Delivery",
          "status": "New"
        }
    }
  }
})
.catch(error => {
  console.log(error)
});
```

getData(dataViewName, payload, context, options)

Retrieves the list of data records in a data view.

> **NOTE:** The terms **data page** and **data view** used in context of this API are interchangeable.

Returns

The response as a Promise.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view from which the list of data records must be retrieved.	☐
payload	object	A query object containing the details of list of columns, filter conditions, and pagination to be retrieved.	☐
context	string	The name of the context where the API is being called.	☐
options	object	The object that contains the properties required to perform additional actions while retrieving the list of data records.	☐

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

Name	Type	Description	Required
query	object	A command to obtain a set of fields satisfying specific conditions, such as, select, sortBy, filter, etc. **NOTE:** This property is applicable only when the **Allow querying any column (Pega connection only)** checkbox is enabled in the data page rule form.	☐
paging	object	An object that obtains a specific number of records from a page. **NOTE:** This property is applicable only when the **Allow querying any column (Pega connection only)** checkbox is enabled in the data page rule form.	☐
dataViewParameters	object	An object whose parameters are configured on the data view or data page. **NOTE:** This property must be provided when the data page has required parameters.	☐

Name	Type	Description	Required
useExtendedTimeout	boolean	The flag that determines if the timeout for the response of the data page must be increased. **NOTE:** This applies only if the data page is sourced by a report definition. • The default value is `false`. • If **useExtendedTimeout** is `true`, the timeout is increased to 45 seconds. • If **useExtendedTimeout** is `false`, the timeout is 10 seconds.	☐

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

Name	Type	Description	Required
skipClearErrorMessages	boolean	The flag that determines if the previously generated error messages should be deleted. **NOTE:** • The default value is `false`.	☐

Name	Type	Description	Required
		• If **skipClearErrorMessages** is `true`, the error messages will not be deleted. • If **skipClearErrorMessages** is `false`, the error messages will be deleted.	
signal	object	The object that is passed through the AbortController interface to abort an existing request when a new request is made.	☐

Usage example

In this example, the API retrieves the first 10 records of employees whose gender is `Male` and whose role is `Software`.

```
const dataViewName = "D_EmployeeList";
const payLoad = {
"dataViewParameters": {
   "dept": "Engineering"
 },
 "query": {
   "distinctResultsOnly": true,
   "filter": {
     "filterConditions": {
       "F1": {
         "comparator": "EQ",
         "ignoreCase": true,
         "lhs": {
```

```
          "field": "Role"
        },
        "rhs": {
          "value": "Software"
        }
      },
      "F2": {
        "comparator": "EQ",
        "ignoreCase": true,
        "lhs": {
          "field": "Gender"
        },
        "rhs": {
          "value": "Male"
        }
      }
    },
    "logic": "F1 AND F2"
  }
  "select": [
    {
      "field": "Name"
    },
    {
      "field": "Role"
    },
    {
      "field": "Gender"
    }
  ]
},
"paging":{
```

```
    "pageNumber":1,
    "pageSize":10
  }
};
const context = "app/primary_1";
PCore.getDataApiUtils().getData(dataViewName, payload, context)
.then(response => {
 // The response of this API is as shown below:
 {
     data: [
       {
         "Name" : "Mark wood",
         "Role" : "Software",
         "Gender" : "Male"
       },
       {
         "Name" : "Gabe Edwards",
         "Role" : "Software",
         "Gender" : "Male"
       }
     ]
     fetchDateTime: "2020-06-29T11:06:24.329Z",
     pageNumber: 1,
     pageSize: 10
 }
})
.catch(error => {
   console.log(error);
});
```

getDataViewMetadata(dataViewName, context, associationFilter, propertyFilter)

Obtains the metadata of a data view.

The metadata contains high level information about the data view, such as the class it applies to, and returns the metadata of the fields from the class and the other classes with which it has associations. For more information on associations, see Associations.

> **NOTE:** The terms **data page** and **data view** used in context of this API are interchangeable.

Returns

The metadata as a Promise.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view whose metadata must be obtained.	☐
context	string	The name of the context where the API is being called.	☐
associationFilter	array.<string>	The list of simple or complex associations from which the fields must be fetched.	☐
propertyFilter	array.<string>	The list of field IDs to which the response should be limited.	☐

Usage example

In this example, the API returns the metadata of the data view whose name is `D_BugList`.

```
const dataViewName = "D_BugList";
const context = "app/primary_1";
PCore.getDataApiUtils().getDataViewMetadata(dataViewName, context);

//The response of this API is as shown below.
{
  "classID": "PegaProjMgmt-Work-Bug",
  "className": "Bug",
  "structure": "List",
  "isQueryable": true,
  "fields": [
    {
      "description": "operator who manager assigns work to",
      "fieldID": "pyAssignedOperator",
      "fieldType": "Identifier",
      "isReadOnly": false,
      "name": "Assigned To",
      "dataType": "Identifier"
    },
    {
      "description": "This property is used to identify the work object's parent
      Backlog and should include the pyID of that work object. In the future,
      it will be derived from the user's data input into UserStoryIDEntry or
      based upon the context of the creation of the item.",
      "displayAs": "pxTextInput",
      "fieldID": "BacklogID",
      "fieldType": "Text (single line)",
```

```
    "isReadOnly": false,
    "name": "Backlog ID",
    "maxLength": 32,
    "dataType": "Text"
  }
 ]
}
```

getListCount(dataViewName, payload, context)

Obtains the number of records in a data view.

> **NOTE:** The terms **data page** and **data view** used in context of this API are interchangeable.

Returns

The number of records as a Promise.

Parameters

Name	Type	Description	Required
dataViewName	string	The name of the data view that contains the records whose count must be obtained.	☐
payload	object	A query object containing the details of list of columns and filter conditions.	☐
context	string	The name of the context where the API is being called.	☐

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

Name	Type	Description	Required
query	object	A command to obtain a set of fields satisfying specific conditions, such as, select, sortBy, filter, etc. **NOTE:** This property is applicable only when the **Allow querying any column (Pega connection only)** checkbox is enabled in the data page rule form.	☐
paging	object	An object that obtains a specific number of records from a page. **NOTE:** This property is applicable only when the **Allow querying any column (Pega connection only)** checkbox is enabled in the data page rule form.	☐
dataViewParameters	object	An object whose parameters are configured on the data view or data page. **NOTE:** This property must be provided when the data page has required parameters.	☐

Usage example

In this example, the API retrieves the number of records of employees whose gender is `Female` and whose role is `Finance`.

```
const dataViewName = "D_EmployeeList";
const payLoad = {
 "dataViewParameters": {
   "dept": "HR"
 },
"query": {
 "distinctResultsOnly": true,
  "filter": {
    "filterConditions": {
      "F1": {
        "comparator": "EQ",
        "ignoreCase": true,
        "lhs": {
          "field": "Role"
        },
        "rhs": {
          "value": "Finance"
        }
      },
     "F2": {
        "comparator": "EQ",
        "ignoreCase": true,
        "lhs": {
          "field": "Gender"
        },
        "rhs": {
          "value": "Female"
```

```
        }
      }
    },
    "logic": "F1 AND F2"
  }
  "select": [
    {
      "field": "pyID"
    }
  ]
 }
};
const context = "app/primary_1";
PCore.getDataApiUtils().getListCount(dataViewName, payload, context)
.then(response => {
 //The response of this API is as shown below:
 {
   fetchDateTime: "2020-06-29T11:06:23.896Z"
   hasMoreResults: false
   resultCount: 4923
 }
})
.catch(error => {
 console.log(error);
});
```

updateCaseEditFieldsData(caseID, changeSet, eTag, context)

Updates the fields of a case.

> **NOTE:** This API is deprecated. Please use the updateCaseEditFieldsData(caseID, changeSet, eTag, context) API in the CaseUtils class instead.

> **NOTE:** To update the fields of a case, you must obtain a lock on the case using the getCaseEditLock(caseID, context) API.

Returns

A Promise that when resolved indicates that the case data is updated successfully.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose fields must be updated.	☐
changeSet	object	The object containing the data to be updated in the fields.	☐
eTag	string	The response header generated when the lock is acquired on a case successfully.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API returns an object indicating that the case data has been successfully updated.

```
const caseID = "METORG-VEHICLEMANAGER-WORK V-7222";
const changeSet = { "METORG-VEHICLEMANAGER-WORK V-7222": { Make: "New Value
```

```
" } };
const eTag = "20200831T114802.686 GMT";
const context = "app/primary_1";
PCore.getDataApiUtils().updateCaseEditFieldsData(caseID, changeSet, eTag, context);
.then(response => {

 // The response of this API is as shown below:
   {
    "data": {
      "caseInfo": {
        "caseTypeID": "MetOrg-VehicleManager-Work-VehiclePurchase",
        "owner": "reactuser",
        "availableActions": [],
        "lastUpdatedBy": "mohaa5",
        "assignments": [],
        "sla": {},
        "createdBy": "reactuser",
        "createTime": "2020-06-08T12:10:08.813Z",
        "urgency": "10",
        "name": "Vehicle Purchase",
        "stages": [],
        "ID": "METORG-VEHICLEMANAGER-WORK V-10001",
        "lastUpdateTime": "2020-09-01T05:52:54.225Z",
        "stageID": "PRIM5",
        "stageLabel": "Request",
        "status": "New"
      }
    },
    "confirmationNote": "Thank you! The next step in this case has been routed appr
opriately."
   }
  })
```

```
.catch(error => {
  console.log(error);
});
```

APIs in the DataPageUtils class

Use the APIs in the DataPageUtils class to retrieve data from data pages.

- **disableCache()**
- **getDataAsync(dataPageName, context, parameters, paging, query, options)**
- **getPageDataAsync(dataPageName, context, parameters, options)**
- **subscribeToDataPageUpdates(subscriptionId, callback, dataPageName, parameters)**
- **unsubscribeToDataPageUpdates(subscriptionId, dataPageName, parameters)**

disableCache()

Disables the caching of the results of data pages that are fetched using APIs from the DataPageUtils class.

Returns

Not applicable.

Parameters

This API does not have parameters.

Usage example

In this example, the API disables the caching of the data page results on the client-side.

```
PCore.getDataPageUtils().disableCache();
```

getDataAsync(dataPageName, context, parameters, paging, query, options)

Obtains results from the specified list type data page.

Returns

The results from the list type data page as a promise.

Parameters

Name	Type	Description	Required
dataPageName	string	The name of the list type data page whose results must be obtained.	☐
context	string	The name of the context from where the API is being called.	☐
parameters	object	The object containing the parameters associated with the data page.	☐
paging	object	The object containing information related to paging for the specified data page.	☐
query	object	The object containing information about the list of columns and filters in the specified data page.	☐
options	object	The JavaScript object containing additional properties for fetching the data page results.	☐

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

Name	Type	Description	Required
invalidateCache	boolean	The flag that indicates whether the cache needs to be invalidated for the current parameter set of the data page passed. **NOTE:** • The default value is `false`. • When **invalidateCache** is set to `true`, the cache is invalidated and the results are fetched from the server.	☐
purgeDataPageCache	boolean	The flag that indicates whether the cache needs to be invalidated for all the parameter sets of the data page passed. **NOTE:** • The default value is `false`. • When **purgeDataPageCache** is set to `true`, the cache is invalidated and the results of the current parameter set are fetched from the server.	☐

Usage example

In this example, the API retrieves the first 10 records of the `D_EmployeeList` data page.

```
const dataViewName = "D_EmployeeList";
const parameters = {
  "dept": "Engineering"
};
const paging = {
  "pageNumber": 1,
  "pageSize": 10
};
const query = {
  "distinctResultsOnly": true,
  "select": [{
      "field": "Name"
    },
    {
      "field": "Role"
    },
    {
      "field": "Gender"
    }
  ]
};
const context = PConnect.getContextName();
PCore.getDataPageUtils().getDataAsync(dataViewName, context, parameters, paging,
 query)
.then(response => {
    // The response of this API is as shown below:
    {
```

```
      data: [{
          "Name": "Mark D",
          "Role": "Software Engineer",
          "Gender": "Male"
        },
        {
          "Name": "Lara",
          "Role": "Electrician",
          "Gender": "Female"
        }
      ]
      fetchDateTime: "2020-06-29T11:06:24.329Z",
      pageNumber: 1,
      pageSize: 10
    }
  })
  .catch(error => {
    console.log(error);
  });
```

getPageDataAsync(dataPageName, context, parameters, options)

Obtains results from the specified page type data page.

Returns

The results from the page type data page as a promise.

Parameters

Name	Type	Description	Required
dataPageName	string	The name of the page type data page whose results must be obtained.	☐
context	string	The name of the context from where the API is being called.	☐
parameters	object	The object containing the parameters associated with the data page.	☐
options	object	The JavaScript object containing additional properties for fetching the data page results.	☐

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

Name	Type	Description	Required
invalidateCache	boolean	The flag that indicates whether the cache needs to invalidated for the current parameter set of the data page passed. **NOTE:** • The default value is `false`. • When **invalidateCache** is set to `true`, the cache is invalidated and the results are fetched from the server.	☐

Usage example

In this example, the API retrieves the record from the `D_FollowedBugsCount` data page.

```
const dataViewName = "D_FollowedBugsCount";
const parameters = {
  ID: "12311",
};
const context = "app/primary_1";
const options = {
  invalidateCache: true,
};
PCore.getDataPageUtils()
  .getPageDataAsync(dataViewName, context, parameters, options)
  .then((response) =>
    // The response of this API is as shown below:
    ({
      MyFollowedBugsCount: 16,
      pzLoadTime: "August 25, 2022 9:50:59 PM UTC",
      pzPageNameHash: "_pa42316787137117pz",
    })
  )
  .catch((error) => {
    console.log(error);
  });
```

subscribeToDataPageUpdates(subscriptionId, callback, dataPageName, parameters)

Subscribes to updates to a data page.

Returns

The Boolean value `true` if the data page is subscribed to successfully.

Parameters

Name	Type	Description	Required
subscriptionId	string	The unique ID assigned to a subscription.	☐
callback	function	The function to be executed whenever a data page is updated.	☐
dataPageName	string	The name of the data page that is being subscribed to.	☐
parameters	object	The object containing parameters associated with the data page. **NOTE:** For client-side invalidation of a data page, the server sends a WebSocket message to the client. • If the `DATAPAGE_UPDATED` WebSocket message only contains the name of a data page, all subscriptions registered with the data page and all subscriptions registered with both the data page and parameter will be invoked. • If the `DATAPAGE_UPDATED` WebSocket message contains the name of a data page and a parameter, all subscriptions	☐

Name	Type	Description	Required
		registered with the data page and specific subscriptions registered with both the data page and parameter will be invoked.	

Usage example

In this example, the API subscribes to updates to the `D_EmployeeList` data page for the `Engineering` department.

```
const subscriptionId = "900150983cd24fb0d6963f7d28e17f72";
const callback = function () {
  const updatedEmployeesList = PCore.getDataPageUtils().getDataAsync("D_Employee
eList", "app/primary_1", {
    "dept": "Engineering"
  });
}
const dataPageName = "D_EmployeeList";
const parameters = {
  "department": "Engineering"
};
const isSubscribed = PCore.getDataPageUtils().subscribeToDataPageUpdates(subscri
ptionId, callback, dataViewName, parameters);
if (isSubscribed) {
  console.info('Subscription successful');
} else {
```

```
  console.info('Subscription failure');
}
```

unsubscribeToDataPageUpdates(subscriptionId, dataPageName, parameters)

Unsubscribes from updates to a data page.

Returns

The Boolean value `true` if the data page is unsubscribed from successfully.

Parameters

Name	Type	Description	Required
subscriptionId	string	The unique ID assigned to a subscription.	☐
dataPageName	string	The name of the data page that is being unsubscribed from.	☐
parameters	object	The object containing parameters associated with the data page.	☐

Usage example

In this example, the API unsubscribes from updates to the `D_EmployeeList` data page for the `Engineering` department.

```
const subscriptionId = "900150983cd24fb0d6963f7d28e17f72";
const dataPageName = "D_EmployeeList";
const parameters = {
  "dept": "Engineering"
};
const isUnsubscribed = PCore.getDataPageUtils().unsubscribeToDataPageUpdates(s
```

```
ubscriptionId, dataPageName, parameters);
if (isUnsubscribed) {
  console.info('Successfully unsubscribed');
} else {
  console.info('Failure during unsubscription');
}
```

APIs in the DataTypeUtils class

Use the APIs in the DataTypeUtils class to retrieve information about data types.

- **getDataPageKeys(dataPageName)**
- **getLookUpDataPage(dataClass)**
- **getLookUpDataPageInfo(dataClass)**
- **getSavableDataPage(dataClass)**

getDataPageKeys(dataPageName)

Obtains the keys for a specified data page.

> **NOTE:** Under the Default Data sources for the data class of the specified data page, ensure that either the Default record lookup data page or Default save data page or Default list data page is configured.

Returns

An array of objects containing the following properties:

- **keyName** - The name of the key associated with the specified data page.
- **isAlternateKeyStorage** - The flag that indicates if alternate key storage is enabled for the specified data page.

- **linkedField** - The property linked to the key.

> **NOTE:** The **linkedField** property can be accessed only if **isAlternateKeyStorage** is `true`.

> **NOTE:** If the data page is not configured for the data class, a null value is returned.

Parameters

Name	Type	Description	Required
dataPageName	string	The name of the data page whose keys must be obtained. **NOTE:** This is the data page configured under the Default Data sources for the specified data class.	☐

Usage example

In this example, the API obtains the keys for the `D_TestSavable` data page.

```
PCore.getDataTypeUtils().getDataPageKeys("D_TestSavable");
```

getLookUpDataPage(dataClass)

Obtains the name of the lookup data page for a specified data class.

> **NOTE:** Under the Default Data sources for the specified data class, ensure that the Default record lookup data page is configured.

Returns

The name of the lookup data page as a string.

> **NOTE:** If the Default record lookup data page is not configured for the data class, a null value is returned.

Parameters

Name	Type	Description	Required
dataClass	string	The name of the data class whose lookup data page name must be obtained.	☐

Usage example

In this example, the API obtains the name of the lookup data page for the `O02LDN-AppReact-Data-Test` data class.

```
PCore.getDataTypeUtils().getLookUpDataPage("O02LDN-AppReact-Data-Test");
```

getLookUpDataPageInfo(dataClass)

Obtains information related to the lookup data page of a specified data class.

> **NOTE:** Under the Default Data sources for the specified data class, ensure that the Default record lookup data page is configured.

Returns

The information related to the lookup data page as an object.

> **NOTE:** If the default record lookup data page is not configured for the data class, a null value is returned.

Parameters

Name	Type	Description	Required
dataClass	string	The name of the data class whose lookup data page information must be obtained.	☐

Usage example

In this example, the API obtains the information of the lookup data page for the `O02LDN-AppReact-Data-Test` data class.

```
PCore.getDataTypeUtils().getLookUpDataPageInfo("O02LDN-AppReact-Data-Test");
```

The information of the lookup data page is returned as the following object:

```
{
  isAlternateKeyStorage:true,
  parameters:{
    'param1_on_DP':'@P .Dependent_Field_in_same_class',
    'Param2_on_DP':'Constant'
  }
}
```

getSavableDataPage(dataClass)

Obtains the name of the savable data page for a specified data class.

> **NOTE:** Under the Default Data sources for the specified data class, ensure that the Default save data page is configured.

Returns

The name of the savable data page as a string.

> **NOTE:** If the Default save data page is not configured for the data class, a null value is returned.

Parameters

Name	Type	Description	Required
dataClass	string	The name of the data class whose savable data page name must be obtained.	☐

Usage example

In this example, the API obtains the name of the savable data page for the `O02LDN-AppReact-Data-Test` data class.

```
PCore.getDataTypeUtils().getSavableDataPage("O02LDN-AppReact-Data-Test");
```

APIs in the EnvironmentInfo class

Use the APIs in the EnvironmentInfo class to retrieve information about the environment that the user is currently logged into.

- **getAccessGroup()**
- **getApplicationLabel()**
- **getApplicationName()**
- **getCaseInstanceListDP()**
- **getCookieComplianceMethod()**
- **getDefaultOperatorDP()**
- **getDefaultPortal()**
- **getEnvironmentKeys()**
- **getKeyMapping(keyValue)**
- **getMaxAttachmentSize()**
- **getOperatorIdentifier()**
- **getOperatorImageInsKey()**
- **getOperatorName()**
- **getOperatorWorkGroup()**
- **getRenderingMode()**
- **getTheme()**
- **getTimeZone()**

- getUseLocale()
- setCookieComplianceMethod(cookieComplianceMethod)
- setLocale(locale)
- setTheme(theme)

getAccessGroup()

Obtains the access group of the user who is logged in currently.

Returns

The access group as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the access group of the user who is logged in currently.

```
const accessGroup = PCore.getEnvironmentInfo().getAccessGroup();
```

getApplicationLabel()

Obtains the label of the application that the user is currently logged into.

Returns

The application label as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the label of the application that the user is currently logged into.

```
const applicationLabel = PCore.getEnvironmentInfo().getApplicationLabel();
```

getApplicationName()

Obtains the name of the application that the user is currently logged into.

Returns

The name of the application as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the name of the application that the user is currently logged into.

```
const applicationName = PCore.getEnvironmentInfo().getApplicationName();
```

getCaseInstanceListDP()

Obtains the name of the data page containing the list of case instances.

Returns

The name of the data page as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the data page containing the list of case instances.

```
const caseInstanceListDP = PCore.getEnvironmentInfo().getCaseInstanceListDP();
```

getCookieComplianceMethod()

Obtains the value of the cookie compliance method or privacy compliance method that is defined on the currently rendered portal.

Returns

The value as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns `default` as the value of the cookie compliance method.

```
const cookieComplianceMethod = PCore.getEnvironmentInfo().getCookieCompliance
Method();
```

getDefaultOperatorDP()

Obtains the name of the data page containing the list of operators.

Returns

The name of the data page as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the data page containing the list of operators.

```
const defaultOperatorDP = PCore.getEnvironmentInfo().getDefaultOperatorDP();
```

getDefaultPortal()

Obtains the name of the default portal assigned to the operator.

Returns

The name of the portal as a string.

Parameters

This API does not have parameters.

Usage example

In this example, if the default portal of the operator is `WebPortal`, the API returns `WebPortal`.

```
PCore.getEnvironmentInfo().getDefaultPortal()
```

getEnvironmentKeys()

Obtains the environment keys that contain values specific to the operator who is currently logged in to the application.

Returns

The environment keys as an array.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the global environment keys that contain values specific to the operator who is currently logged in to the application.

```
const environmentKeys = PCore.getEnvironmentInfo().getEnvironmentKeys();
```

getKeyMapping(keyValue)

Obtains the corresponding Launchpad identifier for a specified Infinity identifer.

Returns

The Launchpad identifier as a string.

> **NOTE:** If there is no corresponding Launchpad identifier for the specified Infinity identifer, a null value is returned.

Parameters

Name	Type	Description	Required
keyValue	string	The Infinity identifer whose corresponding Launchpad identifier must be obtained.	☐

Usage example

In this example, the API obtains the `ID` Launchpad identifier that corresponds to the `pzInsKey` Infinity identifer.

```
PCore.getEnvironmentInfo().getKeyMapping("pzInsKey")
```

getMaxAttachmentSize()

Obtains the system configured size limit in MB for attachments.

> **NOTE:** If the size limit is not configured in the system, then the default value is **5**.

Returns

The size limit as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the system configured size limit for attachments.

```
const maxAttachmentSize = PCore.getEnvironmentInfo().getMaxAttachmentSize();
```

getOperatorIdentifier()

Obtains the identifier of the operator who is currently logged in to the application.

Returns

The operator identifier as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the identifier of the operator who is currently logged in to the application.

```
const operatorIdentifier = PCore.getEnvironmentInfo().getOperatorIdentifier();
```

getOperatorImageInsKey()

Obtains a key that is used to retrieve the image of the operator who is currently logged in to the application.

Returns

The operator image insKey as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains a key, which is used to retrieve the image of the operator who is currently logged in to the application.

```
const operatorImgInsKey = PCore.getEnvironmentInfo().getOperatorImageInsKey();
```

getOperatorName()

Obtains the username of the operator who is currently logged in to the application.

Returns

The username of the operator as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the username of the operator who is currently logged in to the application.

```
const operatorName = PCore.getEnvironmentInfo().getOperatorName();
```

getOperatorWorkGroup()

Obtains the work group of the operator who is currently logged in to the application.

Returns

The name of the work group of the operator as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the work group of the operator who is currently logged in to the application.

```
const operatorWorkGroup = PCore.getEnvironmentInfo().getOperatorWorkGroup();
```

getRenderingMode()

Obtains the mode that the Constellation architecture UI is rendered in.

Constellation architecture UI can be rendered in the following modes:

- When Constellation architecture UI is rendered as a full portal, the rendering mode is **FULL_PORTAL**.
- When Constellation architecture UI is rendered through Web embed channels into web pages, the rendering mode is **EMBED**.
- When Constellation architecture UI is rendered within legacy sections or harnesses, the rendering mode is **HYBRID**.
- When Constellation architecture UI is rendered during authoring, the rendering mode is **PREVIEW**.

Returns

The rendering mode as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the mode that the Constellation architecture UI can be rendered in.

```
const renderingMode = PCore.getEnvironmentInfo().getRenderingMode();
```

getTheme()

Obtains the defined theme overrides of the application that the user is currently logged into.

Returns

The theme override as a JSON object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the defined theme overrides of the application that the user is currently logged into.

```
const theme = PCore.getEnvironmentInfo().getTheme();
```

getTimeZone()

Obtains the defined time zone of the application that the user is currently logged into.

Returns

The defined time zone as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the defined time zone of the application that the user is currently logged into.

```
const timeZone = PCore.getEnvironmentInfo().getTimeZone();
```

getUseLocale()

Obtains the defined locale information of the user who is currently logged into the application.

Returns

The defined locale information as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains the defined locale information of the user who is currently logged into the application.

```
const useLocale = PCore.getEnvironmentInfo().getUseLocale();
```

setCookieComplianceMethod(cookieComplianceMethod)

Sets the value of the cookie compliance method or privacy compliance method for the current portal.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
cookieComplianceMethod	string	The value of the cookie compliance method or privacy compliance method to be set for the current portal.	☐

Usage example

In this example, the API sets `default` as the value of the cookie compliance method for the current portal.

```
PCore.getEnvironmentInfo().setCookieComplianceMethod('default');
```

setLocale(locale)

Changes the user's locale information in a portal to load the localized application content.

> **NOTE:**
>
> - This API must be called when the portal is being loaded.
> - This API changes the user's locale information in a portal, but does not change the user's defined locale information.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
locale	string	The user's locale information that must be changed in the portal.	☐

Usage example

In this example, the API sets the user's locale information to `de_DE` in the portal.

```
PCore.getEnvironmentInfo().setLocale('de_DE');
```

setTheme(theme)

Updates the theme override of the application that the user is currently logged into.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
theme	JSON object	The new theme override of the application.	☐

Usage example

In this example, the API updates the theme override of the application that the user is currently logged into.

```
PCore.getEnvironmentInfo().setTheme(theme);
```

APIs in the ErrorHandler class

Use the APIs in the ErrorHandler class to handle errors.

- **getGenericFailedMessage()**
- **setGenericFailedMessage(message)**

getGenericFailedMessage()

Obtains a generic error message when an error occurs.

Returns

The error message as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains a generic error message.

```
PCore.getErrorHandler().getGenericFailedMessage();
```

setGenericFailedMessage(message)

Sets an error message that is displayed when an error occurs.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
message	string	The error message that must be displayed when an error occurs.	☐

Usage example

In this example, the API sets **Failed to load preview** as the error message to be displayed when an error occurs.

```
PCore.getErrorHandler().setGenericFailedMessage('Failed to load preview');
```

APIs in the Events class

Use the API in the Events class to subscribe to various events.

- **getCaseEvent()**
- **getDataEvent()**

getCaseEvent()

Obtains the case related events that can be subscribed to using the getPubSubUtils() API.

Returns

The name of the event as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the case related event that can be subscribed to.

```
const CaseEvent = PCore.getEvents().getCaseEvent().CASE_CREATED;
```

getDataEvent()

Obtains the data object related events that can be subscribed to using the getPubSubUtils() API.

Returns

The name of the event as a string.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the name of the data object related event that can be subscribed to.

```
const dataEvent = PCore.getEvents().getDataEvent().DATA_OBJECT_CREATED;
```

APIs in the ExpressionEngine class

Use the APIs in the ExpressionEngine class to perform expression-related actions.

- **evaluate(expression, data, options)**

evaluate(expression, data, options)

Evaluates a filter expression on the specified local data object.

Returns

The evaluated filter expression as a Boolean value. The value is `true` when the condition in the expression is met.

Parameters

Name	Type	Description	Required
expression	string	The filter expression that must be evaluated on the specified local data object.	☐
data	object	The local data object that must be filtered. **NOTE:** • The local data object always takes precedence over the data in the Store. If both the local data and the c11nEnv object are passed,only the local data is considered. • This parameter is optional if the c11nEnv object is passed as part of the options object.	☐
options	object	The JavaScript object that contains the c11nEnv object of the property on which the specified filter expression is evaluated.	☐

Name	Type	Description	Required
		NOTE: • This parameter is mandatory if the value of the data object is not provided. • To evaluate the expression on the data in the Store, specify the value of the data object as undefined.	

Usage example

In this example, the API evaluates the specified filter expression on the data object and returns the Boolean value `true`.

```
const data = {
  class: 'MOBILE_PAGE',
  HomeNumber: '456',
  name: 'abc'
};
* const meta = {
  meta: {
    type: 'ScalarList',
    config: {
      ruleClass: 'OSYEB1-TestApp-Data-Address',
      value: '@FILTERED_LIST .Mobile[.HomeNumber != "123"].name'
    }
  },
  options: {
```

```
    pageReference: 'caseInfo.content',
    context: 'app'
  }
};
const pConnect = PCore.createC11nEnv(meta);
const options = {
  pConnect
}
const result = PCore.getExpressionEngine().evaluate(".HomeNumber != '123'", data, o
ptions);
```

APIs in the FeedUtils class

Use the APIs in the FeedUtils class to handle the feeds of a case.

- **deleteMessage(messageID, isReply, replyID, pConnectObj)**
- **editMessage(param)**
- **getFeeds(pulseContext, feedID, feedClass, feedFilters, fetchFeedsCancelTokenSource, pConnectObj, isLoadMore)**
- **getLikedUsers(messageID, pConnectObj)**
- **getMentionSuggestions(mentionProps, pConnectObj)**
- **getMentionTypes(pConnectObj)**
- **getTagSuggestions(tagProps, pConnectObj)**
- **likeMessage(param)**
- **postMessage(pulseContext, message, pConnectObj, attachmentIDs, isReply)**

deleteMessage(messageID, isReply, replyID, pConnectObj)

Deletes a message from a given context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
messageID	string	The ID of the message that needs to be deleted. **NOTE:** Ensure that you provide the `pzInsKey` value of the **messageID**.	☐
isReply	boolean	The flag that determines if you want to delete a message or a reply to a message. **NOTE:** • The default value is `false`. • Set **isReply** to `true` if you want to delete a reply to a message.	☐

Name	Type	Description	Required
		• Set **isReply** to `false` if you want to delete a message.	
replyID	string	The ID of the reply that needs to be deleted. **NOTE:** • This parameter is required if **isReply** is set to `true`. • If **isReply** is set to `false`, pass an empty string as the **replyID**.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API removes the reply whose ID is `PEGASOCIAL M-101` from a message whose ID is `PEGASOCIAL M-100`.

```
PCore.getFeedUtils().deleteMessage('PEGASOCIAL M-100', true, 'PEGASOCIAL M-101',
 getPConnect())
.then(() => {
   // success
}).catch(err => {
```

```
  // errors
});
```

editMessage(param)

Modifies a message associated with the given context.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
param	object	The object that contains the required data to edit a message.	☐

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

Name	Type	Description	Required
feedID	string	The unique identifier of the feed that needs to be edited.	☐
messageID	string	The unique identifier of the message that needs to be edited.	☐
message	string	The new message that replaces the existing message.	☐
context	string	The name of the context where the API is being called.	☐
attachmentIDs	array	The metadata of the attachments that need to be edited along with the message.	☐

Name	Type	Description	Required
isReply	boolean	The flag that determines if you want to edit a message or a reply to a message. **NOTE:** • The default value is `false`. • Set **isReply** to `true` if you want to edit a reply to a message. • Set **isReply** to `false` if you want to edit a message.	☐

Usage example

In this example, the API edits the reply whose ID is `W-104` to the post whose ID `W-103`.

The updated reply is `test reply message`.

```
PCore.getFeedUtils().editMessage({'W-104','W-103','test reply message','app/primary_1', [], true})
.then(() => {
  // success
}).catch(err => {
   // Error handling
});
```

getFeeds(pulseContext, feedID, feedClass, feedFilters, fetchFeedsCancelTokenSource, pConnectObj, isLoadMore)

Obtains the feeds for a given context.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
pulseContext	string	The name of the application context or case context for which the feed must be fetched.	☐
feedID	string	The ID of the feed that must be fetched. **NOTE:** • When the Pulse widget is configured on the Home page, the **feedID** is known as `pyDashboardFeed`. • When the Pulse widget is configured within a case view, the **feedID** is known as `pyCaseFeed`.	☐
feedClass	string	The class associated with the feed to be fetched.	☐

Name	Type	Description	Required
feedFilters	Array.<object>	The list of conditions through which the feed to be fetched is filtered.	☐
fetchFeedsCancelTokenSource	Array.<object>	The list of API requests for fetching the feed. **NOTE:** The latest API request becomes active if the previous API request is pending.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐
isLoadMore	boolean	The flag that determines if the next set of feeds should be loaded. **NOTE:** • The default value is `false`. • Set **isLoadMore** to `true` if you want to load the next set of feeds.	☐

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

Name	Type	Description	Required
id	string	The unique identifier for the feed source.	☐
label	string	The title of the feed source.	☐

Name	Type	Description	Required
disabled	boolean	The flag to be used to deactivate the filter. **NOTE:** • The default value of **disabled** is `false`. • Set **disabled** to `true` to deactivate the filter. • Set **disabled** to `false` to activate the filter.	☐
on	boolean	The flag that conveys if the filter is currently selected. **NOTE:** • The default value of **on** is `false`. • Set **on** to `true` if the filter is currently selected. • Set **on** to `false` if the filter is currently not selected.	☐

Usage example

In this example, the API obtains the feed of a context whose feedID is `pyDashboardFeed`.

```
PCore.getFeedUtils().getFeeds('DATA-PORTAL $EngPMF', 'pyDashboardFeed','class',[{id: 'All', label: 'All', on: false, disabled: false}],[], getPConnect(), true)
.then(feedResponse => {
   // feedResponse array
}).catch(err => {
  // errors
});
```

In this example, the API obtains the feed of a context whose feedID is `pyCaseFeed`.

```
PCore.getFeedUtils().getFeeds('PEGAPROJMGMT-WORK TASK-100', 'pyCaseFeed','class',[{id: 'All', label: 'All', on: false, disabled: false}],[], getPConnect(), true)
.then(feedResponse => {
   // feedResponse array
}).catch(err => {
  // errors
});
```

getLikedUsers(messageID, pConnectObj)

Obtains the list of users who like a message.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
messageID	string	The ID of the message that is liked by the users.	☐

Name	Type	Description	Required
		NOTE: Ensure that you provide the `pzInsKey` value of the **messageID**.	
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the list of users who liked the message with the ID `PEGASOCIAL M-100`.

```
PCore.getFeedUtils().getLikedUsers('PEGASOCIAL M-100', getPConnect())
.then(response => {
  // response array
}).catch(err => {
  // errors
});
```

getMentionSuggestions(mentionProps, pConnectObj)

Obtains the list of options for the selected object that can be mentioned in a Pulse post.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
mentionProps	object	The object that contains the search parameters for obtaining the list of options for the selected object.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the list of cases that can be mentioned in a Pulse post.

```
PCore.getFeedUtils().getMentionSuggestions({searchFor:"case", mentionsType = 'Cases', listSize:5}, getPConnect())
.then(mentionsResponse => {
   // mentionsResponse array
}).catch(err => {
  // errors
});
```

getMentionTypes(pConnectObj)

Obtains the list of available types of objects that can be mentioned in a Pulse post.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the list of available types of objects that can be mentioned in a Pulse post.

```
PCore.getFeedUtils().getMentionTypes(getPConnect())
  .then(response => {
    // response array
  }).catch(err => {
    // errors
  });
```

getTagSuggestions(tagProps, pConnectObj)

Obtains the options suggested for selecting a tag for a given context.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
tagProps	object	The object that contains the search parameters for obtaining the list of tags.	☐

Name	Type	Description	Required
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the options suggested for selecting a tag for the given context.

```
PCore.getFeedUtils().getTagSuggestions({searchFor:"test",listSize:5}, getPConnect())
.then(tagsResponse => {
   // tagsResponse array
}).catch(err => {
  // errors
});
```

likeMessage(param)

Likes or unlikes a message.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
param	object	The object that contains all the required data to like or unlike a specific message.	☐

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

Name	Type	Description	Required
pulseContext	string	The name of the application context or case context for which the feed must be fetched.	☐
likedBy	boolean	The value that determines whether to like or unlike the message. **NOTE:** • Set **likedBy** to `false` if you want to like a message. • Set **likedBy** to `true` if you want to unlike a message.	☐
messageID	string	The ID of the message that needs to be liked or unliked. **NOTE:** Ensure that you provide the `pzInsKey` value of the **messageID**.	☐
isReply	boolean	The flag that determines whether the number of likes must be obtained for a message or a reply to a message. **NOTE:** • The default value is `false`.	☐

Name	Type	Description	Required
		• Set **isReply** to `true` if you want to obtain the number of likes for a reply to a message. • Set **isReply** to `false` if you want to obtain the number of likes for a message.	
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API likes the message whose ID is `PEGASOCIAL M-100`.

```
const payload = {
  pulseContext:'PEGAPROJMGMT-WORK TASK-100',
  likedBy: false,
  messageID:'PEGASOCIAL M-100',
  isReply: false,
  pConnectObj: getPConnect()
}
PCore.getFeedUtils().likeMessage(payload)
.then(() => {
  // success
}).catch(err => {
  // errors
});
```

postMessage(pulseContext, message, pConnectObj, attachmentIDs, isReply)

Posts a message to a specified context.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
pulseContext	string	The name of the application context or case context for which the message must be posted.	☐
message	string	The message that needs to be posted.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐
attachmentIDs	array	The metadata of the attachments that need to be posted along with the message.	☐
isReply	boolean	The flag that determines if you want to post a message or a reply to a message. **NOTE:** • The default value is `false`. • Set **isReply** to `true` if you want to post a reply to a message.	☐

Name	Type	Description	Required
		• Set **isReply** to `false` if you want to post a message.	

Usage example

In this example, the API posts a message along with attachment to the given context.

```
PCore.getFeedUtils().postMessage('DATA-PORTAL','test message', getPConnect(), [{"type":"File","category":"File","fileName":"attachment.png","ID":"459c"}], false)
.then(() => {
  // success
}).catch(err => {
   // Error handling
});
```

APIs in the FieldDefaultUtils class

Use the APIs in the FieldDefaultUtils class to handle the operations related to the default configuration of a field type.

- **getDefaultsforType(type)**
- **setFieldDefault(type, key, value)**
- **updateFieldDefaults(configs)**

getDefaultsforType(type)

Obtains the default configuration of a field type for a component type.

Returns

The default configuration as an object.

Parameters

Name	Type	Description	Required
type	string	The component type whose default configuration of a field type must be obtained.	☐

Usage example

In this example, the API returns the default configuration object for the `Email` component type. An example of the configuration object returned is `{ 'client-validations': true }`.

```
PCore.getFieldDefaultUtils().getDefaultsforType("Email")
```

setFieldDefault(type, key, value)

Sets the default configuration of a field type for a specific component type.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
type	string	The component type whose default configuration of a field type must be set.	☐

Name	Type	Description	Required
key	string	The unique identifier for the default configuration of a field type.	☐
value	any	The value of the configuration key for the component type.	☐

Usage example

In this example, the API sets the default value of `client-validations` to `false` for the `Email` component type.

```
PCore.getFieldDefaultUtils().setFieldDefault("Email","client-validations",false)
```

updateFieldDefaults(configs)

Updates the default configurations of multiple field types.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
configs	object	The object containing the default configurations for multiple component types.	☐

Usage example

In this example, the API disables the default `client-validations` for the `Email` and `Phone` component types.

```
PCore.getFieldDefaultUtils().updateFieldDefaults({
  Email: {
    'client-validations': false
  },
  Phone: {
    'client-validations': false
  }
})
```

APIs in the FieldUtils class

Use the API in the FieldUtils class to handle field-related operations.

- **formatPageReference(referenceList)**

formatPageReference(referenceList)

Updates the array notation path of the target property into a Pega-specific page reference format.

> **NOTE:** The array notation is depicted with **[]** and is a zero-based index, whereas the Pega-specific page reference format is depicted with **()** and is a one-based index.

Returns

The updated reference list as a string.

Parameters

Name	Type	Description	Required
referenceList	string	The array notation path of the target property to be updated.	☐

Usage example

In this example, the API updates the path of the target property into a Pega-specific page reference format.

```
PCore.getFieldUtils().formatPageReference('.Questionsets[0].Questions');
//returns
'.Questionsets(1).Questions'
```

APIs in the FormUtils class

Use the APIs in the FormUtils class to handle form-related cases.

- **clearChangedProperties(context)**
- **getChanges(context)**
- **getEditableFields(context)**
- **getSubmitData(context, options)**
- **isFormValid(context, pageReference)**
- **isStateModified(context)**
- **setCustomValidator(type, validatorFn)**

clearChangedProperties(context)

Deletes the information about data that has changed for a specified context.

> **NOTE:** The changed properties object provides the status of the form during form submission, but it will not delete the Redux state and form fields. After this API is called, if you call the getChanges(context) API to obtain information about the data that has changed for the specified context, an empty object is returned.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The name of the context whose changed data information must be deleted.	☐

Usage example

In this example, the API deletes the information about data that has changed for the `app/primary_1/workarea_1` context.

```
PCore.getFormUtils().clearChangedProperties("app/primary_1/workarea_1");
```

getChanges(context)

Obtains information about the data that has changed for a specified context.

Returns

The information about the changed state as an object.

Parameters

Name	Type	Description	Required
context	string	The name of the context whose changed data information must be obtained.	☐

Usage example

In this example, the API returns information about the data that has changed for the `app/primary_1` context.

```
PCore.getFormUtils().getChanges("app/primary_1");

//The output is as follows:
{
 caseInfo: {
  content: {
     "name": "Optimus",
  }
 },
 pageInstructions: [
  {
   content: {
      "phone1": "+91 9876543210"
   },
   instruction: "INSERT"
   listIndex: 4
   target: ".PhoneNumbers"
```

```
  }
 ]
}
```

getEditableFields(context)

Obtains the details of the editable fields for a specified context.

Returns

The details of the editable fields as an array.

Parameters

Name	Type	Description	Required
context	string	The name of the context whose details of editable fields must be obtained.	☐

Usage example

In this example, the API returns the details of editable fields for the `app/primary_1/workarea_1` context.

```
const editableFields = PCore.getFormUtils().getEditableFields("app/primary_1/workarea_1");
[{
  "name": "caseInfo.content.City",
  "label": "city",
  "type": "textinput"
}]
```

getSubmitData(context, options)

Obtains the submittable data of a context.

Returns

The submittable data of a context as an object.

Parameters

Name	Type	Description	Required
context	string	The name of the context whose submittable data needs to be obtained.	☐
options	object	The JavaScript object that contains properties that provide additional information for obtaining the submittable data.	☐

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

Name	Type	Description	Required
includeDisabledFields	boolean	The flag that determines if disabled fields from the context must be included in the submittable data. **NOTE:** • The default value is `false`. • If **includeDisabledFields** is `true`, disabled fields from	☐

Name	Type	Description	Required
		the context are included in the submittable data. • If **includeDisabledFields** is `true`, disabled fields from the context are not included in the submittable data.	
isTransientContext	boolean	The flag that determines if the submittable data is being obtained from a transient context. **NOTE:** • The default value is `false`. • If **isTransientContext** is `true`, the submittable data is obtained from a transient context. • If **isTransientContext** is `false`, the submittable data is obtained from a non-transient context.	☐

Usage example

In this example, the API returns the submittable data from the `app/primary_1/workarea_1` non-transient context.

```
PCore.getFormUtils().getSubmitData("app/primary_1/workarea_1",{isTransientContext : false});
```

The submittable data is as shown below:

```
{
  caseInfo: {
    content: {
      "name": "Optimus",
    }
  }
}
```

isFormValid(context, pageReference)

Determines if a form is valid based on the values of its fields.

Returns

The Boolean value `true` if the form is valid.

Parameters

Name	Type	Description	Required
context	string	The values of the fields in the form to be validated.	☐
pageReference	string	The path of the embedded page reference property.	☐

Usage example

In this example, the API determines if the specified form is valid.

```
const isFormValid = PCore.getFormUtils().isFormValid("app/primary_1/workarea_1", "
D_Accounts.pxResults[1]");
```

isStateModified(context)

Determines if the Redux State is modified for a specified context.

Returns

The Boolean value `true` if the Redux State is modified for the specified context.

Parameters

Name	Type	Description	Required
context	string	The name of the context.	☐

Usage example

In this example, the API returns the Boolean value `true` if the Redux State is modified for the `app/primary_1/workarea_1` context.

```
PCore.getFormUtils().isStateModified("app/primary_1/workarea_1");
```

setCustomValidator(type, validatorFn)

Registers a custom validator function to the validators object.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
type	string	The type based on which the validation is performed.	☐
validatorFn	function	The function containing the logic to perform the validation.	☐

Usage example

In this example, the API registers a function that performs validation based on the type `email`.

```
PCore.getFormUtils().setCustomValidator('email', () => { // implementation code goes here..  // })
```

APIs in the GenAIAssistantUtils class

Use the APIs in the GenAIAssistantUtils class to handle the GenAI Assistant.

- **createConversation(contextID, assistantID, context, cancelTokenSource)**
- **sendMessage(assistantID, conversationID, message, context)**

createConversation(contextID, assistantID, context, cancelTokenSource)

Initiates a conversation for a specified Pega GenAI Coach™.

Returns

A Promise that resolves to an object containing the details of the conversation with the Pega GenAI Coach.

Parameters

Name	Type	Description	Required
contextID	string	The ID of the context where the conversation must be initiated. **NOTE:** If the context is a Case, ensure that you provide the `pzInsKey` value of the Case.	☐
assistantID	string	The unique identifier of the Pega GenAI Coach with which the conversation must be initiated. **NOTE:** Ensure that you provide the `pxInsName` value of the Pega GenAI Coach.	☐
context	string	The name of the context where the API is being called.	☐
cancelTokenSource	CancelTokenSource[]	The list of tokens corresponding to the API requests that initiate the conversation. **NOTE:** Use `cancelTokenSource` to cancel the current request when a new request is triggered.	☐

Usage example

In this example, the API creates a conversation with a Pega GenAI Coach for a Case.

```
PCore.getGenAIAssistantUtils().createConversation(
    'UPLUS-SAPLUSC11N-WORK-OPPORTUNITY OPP-15001',
    'UPlus-SAPlusC11n-Work-Opportunity-Biz!OpportunityAssistant',
    'app/primary_1',
    [], //Pass the reference of the array
  )
  .then(response => {
    // response
  }).catch(err => {
    // errors
  })
```

sendMessage(assistantID, conversationID, message, context)

Sends a message in an existing conversation with the Pega GenAI Coach™.

Returns

A Promise that resolves to an object containing the response from the Pega GenAI Coach.

Parameters

Name	Type	Description	Required
assistantID	string	The unique identifier of the Pega GenAI Coach.	☐

Name	Type	Description	Required
		NOTE: Ensure that you provide the `pxInsName` value of the Pega GenAI Coach.	
conversationID	string	The unique identifier of the object that contains the details of the conversation.	☐
message	string	The message that must be sent to the Pega GenAI Coach.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API sends a message in an existing conversation with a Pega GenAI Coach.

```
PCore.getGenAIAssistantUtils().sendMessage('UPlus-SAPlusC11n-Work-Opportunity-Biz!OpportunityAssistant','PXCONV-38008','test messge','app/primary_1')
.then(response => {
  // response
}).catch(err => {
  // errors
});
```

APIs in the HeaderProcessor class

Use the APIs in the HeaderProcessor class to perform external header-related actions on service-broker fetch calls.

- **getRegisteredHeaders()**
- **registerHeader(name, value)**
- **unRegisterHeader(name)**

getRegisteredHeaders()

Obtains all external headers that are registered to service-broker fetch calls.

Returns

A JSON object containing the registered external headers and their corresponding values.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns a JSON object containing the registered external headers and their corresponding values.

```
PCore.getRestClient().getHeaderProcessor().getRegisteredHeaders();
```

registerHeader(name, value)

Registers the specified external header to service-broker fetch calls.

Returns

A Boolean value that indicates if the external header has been registered to service-broker fetch calls.

- If the specified external header already exists, the Boolean value `false` is returned.

- If the specified external header is registered successfully, the Boolean value `true` is returned.

Parameters

Name	Type	Description	Required
name	string	The name of the external header that must be registered to service-broker fetch calls.	☐
value	string	The value corresponding to the specified external header.	☐

Usage example

In this example, the API registers the external header with the name `abc` and value `xyz` to service-broker fetch calls and returns the Boolean value `true`.

```
PCore.getRestClient().getHeaderProcessor().registerHeader("abc", "xyz");
```

unRegisterHeader(name)

De-registers the specified external header from service-broker fetch calls.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
name	string	The name of the external header that must be de-registered from service-broker fetch calls.	☐

Usage example

In this example, the API de-registers the external header with the name `abc`.

```
PCore.getRestClient().getHeaderProcessor().unRegisterHeader("abc");
```

APIs in the Initialiser class

Use the APIs in the Initialiser class to initialize default configurations and containers.

- **getBootstrapConfig(restServerUrl, tokenInfo)**
- **init(configObj)**
- **initCoreContainers(options)**

Related concepts

- Initializing the Constellation environment

getBootstrapConfig(restServerUrl, tokenInfo)

Obtains the configuration data from the Pega Infinity server to initialize the Constellation environment.

Returns

The configuration data as an object.

Parameters

Name	Type	Description	Required
restServerUrl	string	The Infinity server URL from which the configuration data is obtained.	☐

Name	Type	Description	Required
tokenInfo	object	The object containing information related to the authorization type and token.	☐

Usage example

In this example, the API returns an object containing the configuration data.

```
const restServerUrl: 'https://example.com/api';
const tokenInfo:{
  token_type: 'Bearer',
  access_token: '##access token##'
};

PCore.getInitialiser().getBootstrapConfig(restServerUrl, tokenInfo);
```

Related concepts

- Initializing the Constellation environment

init(configObj)

Initializes a configuration object that is related to a Constellation application. The configuration object can be obtained from the getBootstrapConfig(restServerUrl, tokenInfo) API.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
configObj	object	The object containing properties to initialize a Constellation application.	☐

Usage example

In this example, the API initializes the provided configuration object.

```
const configObj = {
  restServerUrl: 'https://example.com/api',
  customRendering: false,
  onPCoreReadyCallback: () => {
    console.log('PCore is ready!');
  },
  staticContentServerUrl: 'https://example.com/static',
  authInfo: {}, // (Optional)
  theme: {}, // (Optional)
  renderingMode: 'view',
  appAlias: 'app/C11'
};

PCore.getInitialiser().init(configObj);
```

Related concepts

- Initializing the Constellation environment

initCoreContainers(options)

Creates the required containers to initialize the Constellation application. For more information on containers, see Working with Containers.

Returns

The root container's PConnect object as a prop.

Parameters

Name	Type	Description	Required
options	object	The object containing optional properties that help in initializing the required containers.	☐

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

Name	Type	Description	Required
containerType	string	The type of portal that can be loaded when the the root container and the view container are initialized. • The default value of **containerType** is `single`. • If the value of **containerType** is `single`, single document portals can be loaded. • If the value of **containerType** is `multiple`, multi-document portals can be loaded.	☐

Usage example

In this example, the API initializes the required containers and loads a single document portal.

```
const options = {
  containerType: 'single'
};

PCore.getInitialiser().initCoreContainers(options);
```

Related concepts

- Initializing the Constellation environment

APIs in the LocaleUtils class

Use the APIs in the LocaleUtils class to create, update, and lookup the localization store.

- **getCaseLocaleReference(caseClass, caseName)**
- **getLocaleForRule(localeRuleKey)**
- **getLocaleValue(localeKey, localePath, localeRuleKey, componentName)**
- **getPortalLocaleReference(portal)**
- **getTimeZoneInUse()**
- **loadLocaleResources(localeRefs)**
- **resetLocaleStore()**
- **setLocaleForRule(localeJson, localeRuleKey)**
- **setTimezone(timezone)**

getCaseLocaleReference(caseClass, caseName)

Obtains the locale reference for a case.

Returns

The locale reference as a string.

Parameters

Name	Type	Description	Required
caseClass	string	The class name of the case whose locale reference must be obtained.	☐
caseName	string	The name of the case whose locale reference must be obtained.	☐

Usage example

In this example, the API returns the locale reference for the `Service Request` case whose class is `APP-REACT-WORK`.

```
PCore.getLocaleUtils().getCaseLocaleReference('APP-REACT-WORK', 'Service Request')
```

getLocaleForRule(localeRuleKey)

Obtains the locale JSON for a rule.

Returns

The fields object from the localization store. If the passed key does not exist in the localization store, an undefined value is returned.

Parameters

Name	Type	Description	Required
localeRuleKey	string	The key to the rule whose locale JSON must be obtained. **NOTE:** The default value is `@BASECLASS!GENERIC!PYGENERICFIELDS`.	☐

Usage example

In this example, the API returns a locale JSON for the `WORK-HOME!VIEW!PERSONALINFO` key.

```
PCore.getLocaleUtils().getLocaleForRule("WORK-HOME!VIEW!PERSONALINFO");

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

//The call to the API returns the following JSON object
{
  "fields" : {
```

```
    "First Name" : "Primeiro nome",
    "Last Name"  : "Último nome"
  }
}
```

getLocaleValue(localeKey, localePath, localeRuleKey, componentName)

Obtains the localized value of a string from the localization store.

Returns

The localized value as a string.

Parameters

Name	Type	Description	Required
localeKey	string	The string to be localized.	☐
localePath	string	The locale category in the locale JSON. **NOTE:** The default value is `fields`.	☐
localeRuleKey	string	The key in the localization store. **NOTE:** The default value is `@BASECLASS!GENERIC!PYGENERICFIELDS`.	☐

Name	Type	Description	Required
componentName	string	The name of the component's context that contains the **localeKey**.	☐

Usage example

In this example, the API returns `Primeiro nome` as the localized value.

```
PCore.getLocaleUtils().getLocaleValue("First Name", "", "WORK-HOME!VIEW!PERSONALINFO", "Todo");

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

getPortalLocaleReference(portal)

Obtains the locale reference for a portal.

> **NOTE:** Use this API only when the locale reference is not available in the portal's DX API response.

Returns

The locale reference as a string.

Parameters

Name	Type	Description	Required
portal	string	The ID of the portal whose locale reference must be obtained. **NOTE:** If **portal** is not specified, the current portal is passed.	☐

Usage example

In this example, the API returns `PORTAL!USERPORTAL` as the locale reference for the `UserPortal` portal.

```
PCore.getLocaleUtils().getPortalLocaleReference('UserPortal')
```

getTimeZoneInUse()

Obtains the time zone set in the requestor page. If the time zone is not set in the requestor page, the API obtains the time zone set in the system or browser.

Returns

The time zone value as a string.

Parameters

This API does not have parameters.

Usage example

In this example, if the time zone set in the requestor page is Asia/Calcutta, the API returns `Asia/Calcutta`.

```
PCore.getLocaleUtils().getTimeZoneInUse();
```

loadLocaleResources(localeRefs)

Loads all the locale resources from the locale references if the user locale is different from the base locale.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
localeRefs	array	The list of the locale references to be loaded.	☐

Usage example

In this example, the API loads all the locale resources from the specified locale references if the user locale is different from the base locale.

```
PCore.getLocaleUtils().loadLocaleResources(["WORK-HOME!PAGE!PERSONALINFO", "
WORK-HOME!CASE!FINANCIALSTATUS"]);
```

resetLocaleStore()

Resets the localization store to an empty object.

Returns

Not applicable.

Parameters

This API does not have parameters.

Usage example

In this example, the API resets the localization store to an empty object.

```
PCore.getLocaleUtils().resetLocaleStore();
```

setLocaleForRule(localeJson, localeRuleKey)

Sets the locale JSON for a rule.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
localeJson	object	The locale JSON that must be set for the rule.	☐
localeRuleKey	string	The key to the rule whose locale JSON must be set.	☐

Usage example

In this example, the API sets the locale JSON for the `WORK-HOME!VIEW!PERSONALINFO` key.

```
PCore.getLocaleUtils().setLocaleForRule({ "fields" : {"First Name" : "Primeiro nome","Last Name" : "Último nome"}}, "WORK-HOME!VIEW!PERSONALINFO");

//After the API is called, the localization store has the following structure
{
    "WORK-HOME!VIEW!PERSONALINFO" : {
       "fields" : {
            "First Name" : "Primeiro nome",
            "Last Name"  : "Último nome"
       }
    },
    .
    .
    .
}
```

setTimezone(timezone)

Sets the time zone for the LocaleUtils instance.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
timezone	string	The time zone to be set.	☐

Usage example

In this example, the API sets the time zone to `Asia/Calcutta`.

```
PCore.getLocaleUtils().setTimezone("Asia/Calcutta");
```

APIs in the MashupApi class

Use the APIs in the MashupApi class to create cases or work with views in other environments.

- **createCase(className, targetContext, options)**
- **getCurrentContextAPI(pageReference, targetContext)**
- **getNextWork(targetContext, options)**
- **openAssignment(assignmentId, targetContext, options)**
- **openCase(caseId, targetContext, options)**
- **openPage(pageName, className, targetContext, options)**

createCase(className, targetContext, options)

Creates a case and loads the view into a container.

Returns

An empty Promise.

Parameters

Name	Type	Description	Required
className	string	The name of the class that the created case should belong to.	☐
targetContext	string	The context where the view must be loaded.	☐

Name	Type	Description	Required
		NOTE: If the value of **targetContext** is not provided, the default value is `app`.	
options	object	The JavaScript object containing optional properties that can be used to create a case and load the view.	☐

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

Name	Type	Description	Required
pageName	string	The name of the view that displays the assignment. Use the following values for **pageName**: • `pyEmbedAssignment` - Displays the assignment area, such as, assignment or tasks list. • `pyEmbedAssignmentWithStages` - Displays the assignment area and the case life cycle information above the assignment area. **NOTE:** If the value of **pageName** is not provided, the default value is	☐

Name	Type	Description	Required
		`pyDetails` that displays the assignment in full case view.	
startingFields	object	The JSON object that contains the fields to be set while creating a case. **NOTE:** For more information on setting the fields while creating a case, see Adding fields while creating cases in Constellation DX API .	☐
disableAssignmentHeader	boolean	The flag that determines the visibility of the assignment header. **NOTE:** • The default value is `false`. • If **disableAssignmentHeader** is set to `true`, the assignment header is not visible. • If **disableAssignmentHeader**	☐

Name	Type	Description	Required
		is set to `false`, the assignment header is visible.	

Usage example

In this example, the API creates a case and updates the Store.

```
const options = {
    pageName: "pyEmbedAssignment",
    startingFields: {
        FirstName: "Adam",
        LastName: "Smith",
        Vehicle: {
            Make: "Honda",
            Model: "Accord"
        }
    }
 };

PCore.getMashupApi().createCase('OXJ4P4-CoWin-Work-Feedback', options);
```

getCurrentContextAPI(pageReference, targetContext)

Obtains an object that provides access to the APIs that read or update the state of the context.

Returns

An object that provides access to context APIs.

Parameters

Name	Type	Description	Required
pageReference	string	The reference to the page within the current context.	☐
targetContext	string	The context whose state must be read or updated. **NOTE:** If the value of **targetContext** is not provided, the default value is `app`.	☐

Usage example

In this example, the property of the context object is updated.

```
const contextAPI = PCore.getMashupApi().getCurrentContextAPI("caseInfo.content", "
app/primary_1/workarea_2");
contextAPI.setValue(".CustomerName", "Connor");
```

getNextWork(targetContext, options)

Obtains the assignment that contains the highest priority and loads it into a target context.

Returns

An empty Promise.

Parameters

Name	Type	Description	Required
targetContext	string	The context where the assignment must be loaded. **NOTE:** If the value of **targetContext** is not provided, the default value is `app`.	☐
options	object	The JavaScript object containing the property that can be used to load a specific view.	☐

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

Name	Type	Description	Required
pageName	string	The name of the view that displays the assignment. Use the following values for **pageName**: • `pyEmbedAssignment` - Displays the assignment area, such as, assignment or tasks list. • `pyEmbedAssignmentWithStages` - Displays the assignment area and the case life cycle information above the assignment area.	☐

Name	Type	Description	Required
		NOTE: If the value of **pageName** is not provided, the default value is `pyDetails` that displays the assignment in full case view.	
disableAssignmentHeader	boolean	The flag that determines the visibility of the assignment header. **NOTE:** • The default value is `false`. • If **disableAssignmentHeader** is set to `true`, the assignment header is not visible. • If **disableAssignmentHeader** is set to `false`, the assignment header is visible.	☐

Usage example

In this example, the API loads the assignment with the highest priority into the `workarea` context.

```
PCore.getMashupApi().getNextWork('workarea', { pageName: 'pyEmbedAssignment' }
);
```

openAssignment(assignmentId, targetContext, options)

Loads an assignment into a container.

Returns

An empty Promise.

Parameters

Name	Type	Description	Required
assignmentId	string	The ID of the assignment to be loaded.	☐
targetContext	string	The context where the assignment must be loaded. **NOTE:** If the value of **targetContext** is not provided, the default value is `app`.	☐
options	object	The JavaScript object containing the property that can be used to load a specific view.	☐

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

Name	Type	Description	Required
pageName	string	The name of the view that displays the assignment. Use the following values for **pageName**: • `pyEmbedAssignment` - Displays the assignment area, such as, assignment or tasks list. • `pyEmbedAssignmentWithStages` - Displays the assignment area and the case life cycle information above the assignment area. **NOTE:** If the value of **pageName** is not provided, the default value is `pyDetails` that displays the assignment in full case view.	☐
disableAssignmentHeader	boolean	The flag that determines the visibility of the assignment header. **NOTE:** • The default value is `false`. • If **disableAssignmentHeader** is set to `true`, the assignment header is not visible.	☐

Name	Type	Description	Required
		• If **disableAssignmentHeader** is set to `false`, the assignment header is visible.	

Usage example

In this example, the API opens the assignment and updates the Redux Store.

```
PCore.getMashupApi().openAssignment('Work-Test M-12!Assignment_id');
```

openCase(caseId, targetContext, options)

Loads a case into a target container.

Returns

An empty Promise.

Parameters

Name	Type	Description	Required
caseId	string	The ID of the case to be loaded.	☐
targetContext	string	The target context where the case must be loaded.	☐

Name	Type	Description	Required
		NOTE: If the value of **targetContext** is not provided, the default value is `app`.	
options	object	The JavaScript object containing the property that can be used to load a specific view.	☐

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

Name	Type	Description	Required
pageName	string	The name of the view that displays the case. Use the following values for **pageName**: • `pyEmbedAssignment` - Displays the assignment area, such as, assignment or tasks list. • `pyEmbedAssignmentWithStages` - Displays the assignment area and the case life cycle information above the assignment area. **NOTE:** If the value of **pageName** is not provided, the default value is `pyDetails` that displays the full case view.	☐

Name	Type	Description	Required
disableAssignmentHeader	boolean	The flag that determines the visibility of the assignment header. **NOTE:** • The default value is `false`. • If **disableAssignmentHeader** is set to `true`, the assignment header is not visible. • If **disableAssignmentHeader** is set to `false`, the assignment header is visible.	☐

Usage example

In this example, the API opens the case whose ID is `OPB1HW-SPACETRA-WORK RA-10001`, and updates the Redux Store.

```
PCore.getMashupApi().openCase('OPB1HW-SPACETRA-WORK RA-10001');
```

openPage(pageName, className, targetContext, options)

Loads a page into a container.

Returns

An empty Promise.

Parameters

Name	Type	Description	Required
pageName	string	The name of the page to be loaded.	☐
className	string	The name of the class that the page belongs to.	☐
targetContext	string	The context in which the page must be loaded. **NOTE:** If the value of **targetContext** is not provided, the default value is `app`.	☐
options	object	The JavaScript object containing the property that can be used to specify a view for the default case page.	☐

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

Name	Type	Description	Required
defaultCasePage	string	The name of the page to be used for rendering the case view in any subsequent case actions. Use the following values for **defaultCasePage**:	☐

Name	Type	Description	Required
		• `pyEmbedAssignment` - Displays the assignment area, such as, assignment or tasks list. • `pyEmbedAssignmentWithStages` - Displays the assignment area and the case life cycle information above the assignment area. **NOTE:** If the value of **defaultCasePage** is not provided, the default value is `pyDetails` that displays the full case view.	
disableAssignmentHeader	boolean	The flag that determines the visibility of the assignment header. **NOTE:** • The default value is `false`. • If **disableAssignmentHeader** is set to `true`, the assignment header is not visible. • If **disableAssignmentHeader**	☐

Name	Type	Description	Required
		is set to `false`, the assignment header is visible.	

Usage example

In this example, the API opens the `pyHome` page belonging to the `Data-Portal` class, and updates the Redux Store.

```
PCore.getMashupApi().openPage('pyHome','Data-Portal');
```

APIs in the MessageManager class

Use the APIs in the MessageManager class to access and manipulate messages from the Redux Store.

- **addMessages(config)**
- **clearMessages(config)**
- **getMessages(config)**
- **getValidationErrorMessages(context)**
- **MessagesConfigObject**

addMessages(config)

Associates validation messages to property and associates HTTP messages to context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
config	object	The object containing properties to process the information in the messages. **NOTE:** For more information on the object and its properties, see MessagesConfigObject.	☐

Usage example

In this example, the API adds HTTP messages to a context.

```
addMessages({
  messages: [
    {
      type: 'error',
      message: 'Validation Error'
    }
  ],
  category: 'HTTP'
  context: 'app/primary_2'
});
```

In this example, the API adds validation messages to the property of a context.

```
addMessages({
  messages: [
    {
      type: 'info',
      message: 'Info Message'
    }
  ],
  property: '.lastName'
  context: 'app/primary_2'
});
```

clearMessages(config)

Deletes validation messages from property and deletes HTTP messages from context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
config	object	The object containing properties to process the information in the messages. **NOTE:** For more information on the object and its properties, see MessagesConfigObject.	☐

Usage example

In this example, the API deletes HTTP messages from context.

```
clearMessages({
  type: 'error',
  category: 'HTTP'
  context: 'app/primary_2'
});
```

In this example, the API deletes validation messages from property of a context.

```
clearMessages({
  type: 'error',
  property: '.firstName',
  context: 'app/primary_2'
});
```

getMessages(config)

Retrieves validation messages from property and retrieves HTTP messages from context.

Returns

The messages as an object.

Parameters

Name	Type	Description	Required
config	object	The object containing properties to process the information in the messages.	☐

Name	Type	Description	Required
		NOTE: For more information on the object and its properties, see MessagesConfigObject.	

Usage example

In this example, the API retrieves HTTP messages from context.

```
getMessages({
  type: 'error',
  category: 'HTTP'
  context: 'app/primary_2'
});
```

In this example, the API retrieves validation messages from property of a context.

```
getMessages({
  type: 'error',
  property: '.firstName',
  context: 'app/primary_2'
});
```

getValidationErrorMessages(context)

Retrieves all error messages from a context object.

Returns

The error messages as an array.

Parameters

Name	Type	Description	Required
context	string	The name of the context object whose error messages must be retrieved.	☐

Usage example

In this example, the API retrieves the error messages of the `app/primary_2` object.

```
getValidationErrorMessages('app/primary_2');
```

MessagesConfigObject

Use the properties in this JavaScript object to process the information in the messages from the Redux Store.

Properties

Name	Type	Description	Required
type	string	The classification of the message. The value of type can be `error`, `info`, or `success`.	☐
property	string	The name of the property that must be bound to a component.	☐
pageReference	string	The relative path of the property that must be bound to a component.	☐
category	string	The value that determines the position of the message.	☐

Name	Type	Description	Required
		The value of category can be `HTTP`, `PAGE`, or `Property`.	
context	string	The name of the context where the message is displayed. For example, `app` or `app/primary_2` or `app/primary_2/workarea_1`	☐

APIs in the MessagingServiceManager class

Use the APIs in the MessagingServiceManager class to interact with the Constellation Messaging Service.

- **subscribe(filter, messageHandler, contextName, id)**
- **unsubscribe(id)**

subscribe(filter, messageHandler, contextName, id)

Subscribes to the messaging service and forwards messages to subscribers based on a specified filter criteria.

Returns

The user-assigned or auto-generated subscription ID as a string.

Parameters

Name	Type	Description	Required
filter	object	The object containing the filter criteria.	☐

Name	Type	Description	Required
messageHandler	function	The callback function that needs to be invoked.	☐
contextName	string	The name of the context from where the API is being called.	☐
id	string	The user-assigned or auto-generated unique identifier of the subscriber to whom the messages can be forwarded.	☐

Usage example

In this example, the API subscribes to the messaging service and forwards the messages to subscribers based on the specified filter criteria.

```
PCore.getMessagingServiceManager().subscribe({matcher: "interaction"}, message =
> {
 // Perform message execution here
}));
```

unsubscribe(id)

Removes the handler from its subscriptions and disconnects the web socket from the messaging service.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
id	string	The unique identifier of the subscription from which the handler must be removed.	☐

Usage example

In this example, the API removes the handler from its subscriptions and disconnects the web socket from the messaging service.

```
const subId = PCore.getMessagingServiceManager().subscribe({matcher: "interaction
"}, message => {
 // Perform message execution here
}));
PCore.getMessagingServiceManager().unsubscribe(subId);
```

APIs in the MetadataUtils class

Use the APIs in the MetadataUtils class to update and retrieve rule metadata from the Store.

- **getDataPageMetadata(dataPageName)**
- **getEmbeddedPropertyMetadata(propertyName, currentClassID, embeddedType, categoryPath)**
- **getFieldParameters(propertyName, classID)**
- **getGenAICoach(genAICoachName, classID)**
- **getInsight(insightId)**
- **getPersonalizationMetadata(personalizationId)**

- getPropertyMetadata(propertyName, currentClassID)
- getResolvedFieldMetadata(propertyReference, context)
- isViewExists(viewName, classID)
- resolveView(name)

getDataPageMetadata(dataPageName)

Obtains the metadata of a data page.

Returns

The metadata of the data page as an object. If the data page does not exist, a null value is returned.

Parameters

Name	Type	Description	Required
dataPageName	string	The name of the data page whose metadata must be obtained.	☐

Usage example

In this example, the API obtains the metadata of the `D_pyMyWorkList` data page.

```
PCore.getMetadataUtils().getDataPageMetadata("D_pyMyWorkList");
```

The metadata is as shown below:

```
{
  "classID":"Assign-Worklist",
  "mode":"readonly",
```

```
    "isSearchable":false,
    "isQueryable":true,
    "structure":"list",
    "refreshStrategy":{
      "type":"reloadOncePerInteraction"
    },
    "isWorkObject":false,
    "isAssignObject":true
  }
```

getEmbeddedPropertyMetadata(propertyName, currentClassID, embeddedType, categoryPath)

Obtains the metadata of an embedded property's leaf node from the Store.

Returns

The metadata of the embedded property's leaf node as an object. If the property name or class ID are not provided, or if property metadata or Store does not exist, a null value is returned.

Parameters

Name	Type	Description	Required
propertyName	string	The name of the property whose leaf node metadata must be obtained.	☐
currentClassID	string	The unique identifier of the class to which the top level property belongs.	☐
embeddedType	string	The type of the embedded field to which the current leaf node belongs.	☐
categoryPath	any[]	The category to which the current leaf node belongs.	☐

Usage example

In this example, the API obtains the metadata of the `City` leaf node, which needs to be resolved from the `c-1` top level class.

```
//{Customer: [{ classID: 'c-1', pageClass: 'c-2', label:'Customer' }],
// Address: [{ classID: 'c-2', pageClass: 'c-3', label:'Address' }],
// City: [{ classID: 'c-3', label:'City' }]}

PCore.getMetadataUtils().getEmbeddedPropertyMetadata('Customer.Address.City', 'c-1');
```

The metadata is as shown below:

```
{ classID: 'c-3', category: 'Customer.Address' }
```

getFieldParameters(propertyName, classID)

Obtains the parameters of a property after checking if it has a data source.

Returns

The parameters of a property as an object. If the data source of the property does not have parameters or if the property is not associated with a data source, a null value is returned.

Parameters

Name	Type	Description	Required
propertyName	string	The name of the property whose parameters must be obtained.	☐

Name	Type	Description	Required
classID	string	The unique identifier of the class to which the property belongs.	☐

Usage example

In this example, the API obtains the parameters of the property `TestProperty` that belongs to the class `c-1`.

```
PCore.getMetadataUtils().getFieldParameters("TestProperty", "c-1");
```

The parameters are as shown below:

```
{
  "param1":"test"
}
```

getGenAICoach(genAICoachName, classID)

Obtains the metadata of a GenAI coach.

Returns

The metadata as an object. If the GenAI coach does not exist in the Store or if the GenAI coach is not associated with the specified class, a null value is returned.

Parameters

Name	Type	Description	Required
genAICoachName	string	The GenAI coach whose metadata must be obtained.	☐

Name	Type	Description	Required
classID	string	The unique identifier of the class associated with the GenAI coach.	☐

Usage example

In this example, the API obtains the metadata of the `pyGenAICoachDefault` GenAI coach associated with the `Work-` class.

```
PCore.getMetadataUtils().getGenAICoach("pyGenAICoachDefault", "Work-");
```

getInsight(insightId)

Obtains the metadata of an insight.

Returns

The metadata of the insight as an object. If the insight is not present in the Store, a null value is returned.

Parameters

Name	Type	Description	Required
insightId	string	The unique identifier of the insight whose metadata must be obtained.	☐

Usage example

In this example, the API obtains the metadata of the `InsightId` insight.

```
PCore.getMetadataUtils().getInsight("InsightId");
```

The metadata is as shown below:

```
{
  "pyID":"InsightId",
  "pyContent":"{}",
  "pyClassLabel":"Test"
}
```

getPersonalizationMetadata(personalizationId)

Obtains the metadata of a personalization.

Returns

The metadata of the personalization as an object. If the personalization does not exist, a null value is returned.

Parameters

Name	Type	Description	Required
personalizationId	string	The unique identifier of the personalization whose metadata must be obtained.	☐

Usage example

In this example, the API obtains the metadata of the `PersonalizationSample` personalization.

```
PCore.getMetadataUtils().getPersonalizationMetadata("PersonalizationSample");
```

The metadata is as shown below:

```
{
  "defaultPersonalization":"",
  "allPersonalizations": []
}
```

getPropertyMetadata(propertyName, currentClassID)

Obtains the metadata of a property from the Store.

Returns

The metadata from the Store as an object. If the property is not specified, or the property's metadata or the Store does not exist, a null value is returned.

Parameters

Name	Type	Description	Required
propertyName	string	The name of the property whose metadata must be obtained.	☐
currentClassID	string	The unique identifier of the class to which the property belongs.	☐

Usage example

In this example, the API obtains the metadata of the `apartment` property under the `c-1` class.

```
PCore.getMetadataUtils().getPropertyMetadata('apartment', 'c-1');
```

The metadata is as shown below:

```
{
  classID: 'c-1',
  type: 'text'
}
```

getResolvedFieldMetadata(propertyReference, context)

Obtains the resolved field metadata of a property.

Returns

The resolved field metadata as an object.

Parameters

Name	Type	Description	Required
propertyReference	string	The complete reference of the property whose field metadata must be obtained.	☐
context	string	The context in which the property is present.	☐

Usage example

In this example, the API obtains the resolved field metadata of the `FirstName` property under the `app/primary` context.

```
//fieldMetadata
//  {
//    "FirstName":[
//      {
//        "classID":"Address",
//        "type":"text",
```

```
//          "maxLength":10,
//          "additionalInformation":"@PARAGRAPH Instruction"
//        }
//      ]
//   }
const propertyRef = "caseInfo.content.FirstName";
const context = "app/primary";
const resolvedFieldMetadata = PCore.getMetadataUtils().getResolvedFieldMetadata(
propertyRef, context);
```

The resolved field metadata is as shown below:

```
{
  "classID":"Address",
  "type":"text",
  "maxLength":10,
  "additionalInformation":"<p>Hello, how are you</p>"
}
```

isViewExists(viewName, classID)

Determines if the view generated for a class exists in the Store.

Returns

The Boolean value `true` if the view generated for a class exists in the Store.

Parameters

Name	Type	Description	Required
viewName	string	The name of the generated view whose existence in the Store needs to be determined.	☐
classID	string	The unique identifier of the class for which the view was generated.	☐

Usage example

In this example, the API returns the Boolean value `true` if the view `View1` generated for the class `O02LDN-CosmoReact-Work-Test` exists in the Store.

```
PCore.getMetadataUtils().isViewExists("View1", "O02LDN-CosmoReact-Work-Test");
```

resolveView(name)

Obtains the metadata of a view.

Returns

The metadata of a view as an object.

Parameters

Name	Type	Description	Required
name	string	The name of the view whose metadata must be obtained.	☐

Usage example

In this example, the API obtains the metadata of the view `View1`.

```
PCore.getMetadataUtils().resolveView("View1");
```

The metadata is as shown below:

```
{
                "name": "View1",
                "type": "View",
                "config": {
                    "type": "landingpage",
                    "icon": "pi pi-home-solid",
                    "title": "@ENV APPLICATION_DESC",
                    "template": "WideNarrowPage",
                    "ruleClass": "c-1",
                    "localeReference": "@LR C-1!PAGE!VIEW1",
                    "enableGetNextWork": false
                },
                "children": [
                    {
                        "name": "A",
                        "type": "Region",
                        "children": [
                            {
                                "type": "Pulse",
                                "config": {
                                    "label": "@L Pulse",
                                    "messageIDs": "@P pulse.messageIDs"
                                }
                            }
```

```
                ]
            }
        ],
        "classID": "c-1"
}
```

APIs in the NavigationUtils class

Use the APIs in the NavigationUtils class to maintain the state of UI components.

- **getComponentCache(key)**
- **getComponentState(key)**
- **getUserSettings(path)**
- **init()**
- **removeComponentState(key)**
- **resetComponentCache(key)**
- **setComponentCache(key, value, options)**
- **setComponentState(key, state)**
- **setUserSettings(path, value)**

getComponentCache(key)

Obtains the cached value of a specified key.

Returns

The cached value of the key.

Parameters

Name	Type	Description	Required
key	string	The unique identifier whose cached value must be obtained.	☐

Usage example

In this example, the API obtains the cached value of the `searchandselect` key.

```
PCore.getNavigationUtils().getComponentCache("searchandselect");
```

getComponentState(key)

Obtains the state of a specified UI component.

Returns

The state of the UI component as a JSON object.

Parameters

Name	Type	Description	Required
key	string	The ID of the UI component whose state must be obtained.	☐

Usage example

In this example, the API obtains the state of the UI component whose key is `on8ttl-c11ngall-work-d-2001-caseview`.

```
PCore.getNavigationUtils().getComponentState("on8ttl-c11ngall-work-d-2001-caseview");
```

getUserSettings(path)

Obtains the value of a specified property in the `userSettings` attribute.

Returns

The value of the property.

Parameters

Name	Type	Description	Required
path	string	The location of the property whose value must be obtained.	☐

Usage example

In this example, the API returns the value of the `'prop1'` property in the `userSettings` attribute.

```
PCore.getNavigationUtils().getUserSettings('prop1');
```

init()

Initializes the `userSettings` attribute of the NavigationUtils class.

Returns

Not applicable.

Parameters

This API does not have parameters.

Usage example

In this example, the API initializes the `userSettings` attribute of the NavigationUtils class.

```
PCore.getNavigationUtils().init();
```

removeComponentState(key)

Deletes the state of a specified UI component.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
key	string	The unique ID of the UI component whose state must be deleted.	☐

Usage example

In this example, the API deletes the state of the UI component whose key is `on8ttl-c11ngall-work-d-2001-caseview`.

```
PCore.getNavigationUtils().removeComponentState("on8ttl-c11ngall-work-d-2001-caseview")
```

resetComponentCache(key)

Deletes the specified key from the cache.

Returns

The Boolean value `true` if the key is deleted from the cache.

Parameters

Name	Type	Description	Required
key	string	The unique identifier that must be deleted from the cache.	☐

Usage example

In this example, the API deletes the `uniquekeyacrosstheapp` key from the cache.

```
PCore.getNavigationUtils().resetComponentCache("uniquekeyacrosstheapp");
```

setComponentCache(key, value, options)

Caches the assigned value of a specified key in the memory.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
key	string	The unique identifier whose assigned value must be cached.	☐

Name	Type	Description	Required
value	any	The value of the key that must be cached.	☐
options	object	The JavaScript object that contains properties to manage the key.	☐

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

Name	Type	Description	Required
clearOnCancelForContext	string	The name of the context that is used to delete the key. When the given context is removed, the key is also deleted.	☐

Usage example

In this example, the API caches the value assigned to the `uniquekeyacrosstheapp` key.

```
const value = {
  search: {
    filter1: 'name'
  }};
const options =  {
    clearOnCancelForContext: 'app/primary_1/workarea_1
  };
PCore.getNavigationUtils().setComponentState("uniquekeyacrosstheapp", value, options);
```

setComponentState(key, state)

Updates the state of a specified UI component. The updated state is stored in a browser session.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
key	string	The ID of the UI component whose state must be updated.	☐
state	object	The data that must be set as the state of the UI component.	☐

Usage example

In this example, the API updates the state of a UI component whose key is `on8ttl-c11ngall-work-d-2001-caseview`.

```
PCore.getNavigationUtils().setComponentState("on8ttl-c11ngall-work-d-2001-caseview", { active: 0});
```

setUserSettings(path, value)

Specifies or updates values of properties in the `userSettings` attribute.

Returns

A promise that resolves to the updated user settings as an object.

Parameters

Name	Type	Description	Required
path	string	The location of the property whose value must be updated.	☐
value	any	The value that must be assigned to the property.	☐

Usage example

In this example, the API sets the value of the `'prop1'` property to true in the `userSettings` attribute.

```
PCore.getNavigationUtils().setUserSettings('prop1', true);
```

APIs in the PersonalizationUtils class

Use the APIs in the PersonalizationUtils class to manage the personalization instances of a list component.

- **createPersonalization(listID, personalizationID, personalizedState)**
- **deletePersonalization(listID, personalizationID)**
- **fetchPersonalizations(listID)**
- **updatePersonalization(listID, personalizationID, personalizedState)**

createPersonalization(listID, personalizationID, personalizedState)

Creates a new personalization instance for a list component.

Returns

The personalization ID as a Promise.

Parameters

Name	Type	Description	Required
listID	string	A unique ID referencing a list component. The length of the ID should be limited to 32 characters.	☐
personalizationID	string	A temporary unique ID which will be replaced by the actual ID returned by this API.	☐
personalizedState	object	An object containing information about the personalization state.	☐

Usage example

In this example, the API creates a personalization instance and returns a personalization ID.

```
const listId = "443533r555";
const personalizationId = "L_343456";
const personalizedState = {
  "name" : "Open bugs",
  "markAsDefault" : true,
  "personalizationState" : "{ filter : 'pyStatusWork = Open'}"
};

PCore.getPersonalizationUtils(listId).createPersonalization(personalizationId, personalizedState).then((response)=>{
```

```
  const personalizationId = response;
});
```

deletePersonalization(listID, personalizationID)

Deletes a personalization instance of a list component and returns the status.

Returns

The status as a Promise.

Parameters

Name	Type	Description	Required
listID	string	A unique ID referencing a list component. **NOTE:** The length of the ID should be limited to 32 characters.	☐
personalizationID	string	A unique ID that references the personalization instance to be deleted.	☐

Usage example

In this example, the API deletes a personalization instance and returns the status.

```
const listId = "443533r555";
const personalizationId = "L_343456";
PCore.getPersonalizationUtils(listID).deletePersonalization(personalizationId);
```

fetchPersonalizations(listID)

Retrieves the list of personalization instances for a list component and returns the list as a Promise.

> **NOTE:** The API's response also contains a `defaultPersonalization` flag which contains the default personalization ID. If the default personalization ID does not exist, the `defaultPersonalization` flag is empty.

Returns

The list of personalization instances as a Promise.

Parameters

Name	Type	Description	Required
listID	string	A unique ID referencing a list component. **NOTE:** The length of the ID should be limited to 32 characters.	☐

Usage example

In this example, the API retrieves the list of personalization instances for the list component with ID `443533r555`.

```
const listId = "443533r555";
PCore.getPersonalizationUtils("443533r555").fetchPersonalizations();
```

The response structure is as shown below:

```
{
  defaultPersonalization : "L_234322",
  allPersonalizations : [
    {
      "name" : "Open bugs",
      "personalizationId" : "L_345643",
      "personalizationState" : "{ filter : 'pyStatusWork = Open'}"
    },
    {
      "name" : "Resolved bugs",
      "personalizationId" : "L_234322",
      "personalizationState" : "{ filter : 'pyStatusWork STARTS_WITH Resolved'}"
    }
  ]
}
```

updatePersonalization(listID, personalizationID, personalizedState)

Updates a personalization instance of a list component.

Returns

The status as a Promise.

Parameters

Name	Type	Description	Required
listID	string	A unique ID referencing a list component.	☐

Name	Type	Description	Required
		NOTE: The length of the ID should be limited to 32 characters.	
personalizationID	string	A unique ID that references the personalization instance to be updated.	☐
personalizedState	object	An object containing information about the personalization state.	☐

Usage example

In this example, the API updates a personalization instance of a list component and returns the status.

```
const listId = "443533r555";
const personalizationId = "L_343456";
const personalizedState = {
  name : "Resolved bugs",
  markAsDefault : false,
  personalizationState : "{ filter : 'pyStatusWork STARTS_WITH Resolved'}"
};
PCore.getPersonalizationUtils(listId).updatePersonalization(personalizationId, personalizedState);
```

APIs in the PubSubUtils class

Use the APIs in the PubSubUtils class to publish and subscribe to events.

- **cleanContextSubscribers(contextName)**
- **publish(eventType, payload)**

- **subscribe(eventType, subscriptionItem, subscriptionItemName, subscribeOnce, contextName)**
- **subscribeOnce(eventType, subscriptionItem, subscriptionItemName)**
- **unsubscribe(eventType, subscriptionItemName, contextName)**

cleanContextSubscribers(contextName)

Deletes the subscription items of all events under the given context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
contextName	string	The name of the context containing the events whose subscription items must be deleted. **NOTE:** The default value of **contextName** is `app`.	☐

Usage example

In this example, the API deletes the subscription items of all the events under the `app/primary_1` context.

```
PCore.getPubSubUtils().cleanContextSubscribers("app/primary_1");
```

publish(eventType, payload)

Invokes all items subscribed to a specific type of event and passes the specified payload to the subscribed items.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
eventType	string	The type of event whose subscribed events must be invoked.	☐
payload	object	The information that must be passed to the subscribed items.	☐

Usage example

In this example, the API invokes all items subscribed to the event type and passes the payload to the items.

```
PCore.getPubSubUtils().publish(eventType,payload);
```

subscribe(eventType, subscriptionItem, subscriptionItemName, subscribeOnce, contextName)

Notifies a subscription item whenever a specific type of event occurs in a specific context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
eventType	string	The type of event that occurs. **NOTE:** To view the out-of-the-box events, see List of OOTB events.	☐
subscriptionItem	function	The callback function that must be invoked when a specific type of event occurs.	☐
subscriptionItemName	string	A unique ID or name assigned by the user to identify the subscription item.	☐
subscribeOnce	boolean	The flag that determines if the subscription item should be notified only once. **NOTE:** • The default value is `false`. • Set **subscribeOnce** to `true` if you want to notify the subscription item only once.	☐
contextName	string	The name of the context containing the event whose subscription item must be notified.	☐

Name	Type	Description	Required
		NOTE: The default value of **contextName** is `app/primary_1`.	

Usage example

In this example, the API notifies the `createStageCancelAlert` function whenever an event of type `showCancelAlert` occurs.

```
PCore.getPubSubUtils().subscribe("showCancelAlert",()=>{},"createStageCancelAlert",false,"app/primary_1");
```

subscribeOnce(eventType, subscriptionItem, subscriptionItemName)

Notifies the subscription item only once when a specific type of event occurs.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
eventType	string	The type of event that occurs.	☐
subscriptionItem	function	The callback function that must be invoked once when a specific type of event occurs.	☐

Name	Type	Description	Required
subscriptionItemName	function	The unique name or ID used to identify the subscription item.	☐

Usage example

In this example, the API notifies the `createStageCancelAlert` function only once when an event of type `showCancelAlert` occurs.

```
PCore.getPubSubUtils().subscribeOnce("showCancelAlert",()=>{},"createStageCancelAlert");
```

unsubscribe(eventType, subscriptionItemName, contextName)

Unsubscribes a subscription item belonging to a specific type of event.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
eventType	string	The type of event.	☐
subscriptionItemName	function	The unique ID or name used to identify the subscription item.	☐
contextName	string	The name of the context containing the event whose subscription item must be unsubscribed.	☐

Name	Type	Description	Required
		NOTE: The default value of **contextName** is `app/primary_1`.	

Usage example

In this example, the API unsubscribes the `createStageCancelAlert` function belonging to the event of type `showCancelAlert`.

```
PCore.getPubSubUtils().unsubscribe("showCancelAlert","createStageCancelAlert","app/primary_1");
```

APIs in the RelatedCasesApi class

Use the APIs in the RelatedCasesApi class to handle the related cases of a case.

- **addRelatedCases(caseID, relatedCases, context)**
- **getRelatedCases(caseID, context)**
- **removeRelatedCase(caseID, relatedCaseID, context)**

addRelatedCases(caseID, relatedCases, context)

Relates several cases to a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case to which several cases can be related. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
relatedCases	array	The list of case IDs that must be related to a case.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API relates two cases whose IDs are `T-200` and `T-201` to the case whose ID is `W-102`.

```
const casesArray =  [{ ID : 'ORG-MYAPP-WORK T-200'}, {ID : 'ORG-MYAPP-WORK T-201'
}]
PCore.getRelatedCasesApi().addRelatedCases('ORG-MYAPP-WORK W-102', casesArra
y, 'app/primary_1')
.then(() => {
 // success
}).catch(err => {
  // Error handling
});
```

getRelatedCases(caseID, context)

Obtains the related cases of a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose related cases must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API obtains the related cases of a case whose ID is `W-102`.

```
PCore.getRelatedCasesApi().getRelatedCases('ORG-MYAPP-WORK W-102', 'app/primary_1')
        .then(relatedCases => {
        // relatedCases array
        }).catch(err => {
```

```
    // errors
    });
```

removeRelatedCase(caseID, relatedCaseID, context)

Removes a related case from its relationship with a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case from whose relationship the related case must be removed. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
relatedCaseID	string	The ID of the case that must be removed from the relationship. **NOTE:** Ensure that you provide the `pzInsKey` value of the **relatedCaseID**.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API removes the case whose ID is `T-200` from its relationship with the case whose ID is `W-102`.

```
PCore.getRelatedCasesApi().removeRelatedCase('ORG-MYAPP-WORK W-102', 'ORG-MYAPP-WORK T-200', 'app/primary_1')
.then(() => {
   // success
}).catch(err => {
  // errors
});
```

APIs in the RestClient class

Use the APIs in the RestClient class to utilize the service broker to manage REST API calls.

- **getCancelTokenSource()**
- **getHeaderProcessor()**
- **invokeCustomRestApi(endpointUrl, config, context)**
- **invokeRestApi(routeKey, restAPIPayload, context, options)**
- **isRequestCanceled(err)**
- **RestApiConfigObject**
- **RestAPIPayload**

getCancelTokenSource()

Obtains a cancel token source object. The cancel token source object can cancel a request by passing a token as a signal to it.

Returns

The cancel token source as an object.

Parameters

This API does not have parameters.

Usage example

In this example, the API obtains a cancel token source object and uses it to cancel a request.

```
const { getCancelTokenSource } = PCore.getRestClient();
const cancelTokenSource = getCancelTokenSource();
cancelTokenSource.cancel();
```

getHeaderProcessor()

Obtains an entry point to the HeaderProcessor object that contains APIs to perform external header-related actions on service-broker fetch calls.

To view the APIs in the HeaderProcessor class, see APIs in the HeaderProcessor class.

Returns

The HeaderProcessor object.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns the HeaderProcessor object.

```
PCore.getRestClient().getHeaderProcessor();
```

invokeCustomRestApi(endpointUrl, config, context)

Invokes a custom REST API using an endpoint URL.

The custom REST APIs can include external APIs or Pega APIs.

Returns

The custom REST API as a promise.

Parameters

Name	Type	Description	Required
endpointUrl	string	The URL of the REST endpoint. **NOTE:** The URL can be relative (`'/api/dev/v1/insights'`) or absolute (`'https://cs.rpega.com/prweb/api/dev/v1/insights'`).	☐
config	RestApi Config Object	The object containing the information required to invoke the custom REST API. For more details, see RestApiConfigObject.	☐
context	string	The name of the context where the API is being called.	☐

Usage example

In this example, the API calls a custom REST API to get Feed messages.

```
const { invokeCustomRestApi } = PCore.getRestClient();
invokeCustomRestApi("/api/dev/v1/insights", {
  method: "GET",
  body: {},
  headers: {},
})
.then(() => {
  // handle the response
})
.catch((error) => {
  // handle the error
});
```

invokeRestApi(routeKey, restAPIPayload, context, options)

Invokes a specific Pega REST API using a route key.

Returns

The Pega REST API as a promise.

Parameters

Name	Type	Description	Required
routeKey	string	The identifier for the Pega REST API to be invoked.	☐

Name	Type	Description	Required
restAPIPayload	RestAPIPayload	The object containing the options to be provided for the Pega REST API to be invoked. For more details, see RestAPIPayload.	☐
context	string	The name of the context where the API is being called.	☐
options	object	The object containing additional information for invoking a REST endpoint.	☐

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

Name	Type	Description	Required
cancelContext	string	The context used to maintain the active request count.	☐
doNotMergeHttpMessagesForStatusCode	string	The error status code that is used to stop merging the error messages.	☐
includeRemoteSystemIdIfPresent	boolean	The flag that determines if details of the remote system must be included in the request. **NOTE:** • The default value is `false`. • If **includeRemoteSystemIdIfPresent** is `true`, the details	☐

Name	Type	Description	Required
		of the remote system are included in the request. • If **includeRemoteSystemIdIfPresent** is `false`, the details of the remote system are excluded from the request.	

Usage example

In this example, the API calls a Pega REST API to get Feed messages.

```
const { invokeRestApi } = PCore.getRestClient();
const cancelTokenSource = getCancelTokenSource();
invokeRestApi('getFeedMessages', {
  queryPayload: {
    filterForContext: 'DATA-PORTAL $SpaceTra',
    filterByContext: 'context'
  },
  body: {},
  headers: {},
  // passing cancel token so that we can cancel the request using cancelTokenSource
  cancelTokenSource: cancelTokenSource.token
})
.then(() => {
  // handle the response
})
.catch((error) => {
  // handle error
```

```
  if(isRequestCanceled(error)) {
    // handle the canceled request using cancelTokenSource.cancel();
  }
});
```

isRequestCanceled(err)

Uses a cancel token source object to determine if a request has been canceled.

Returns

The Boolean value `true` if the request is canceled.

Parameters

Name	Type	Description	Required
err	object	The error object received when the request is canceled.	☐

Usage example

In this example, the API uses a cancel token source object to determine if a request has been canceled.

```
const { isRequestCanceled } = PCore.getRestClient();
if(isRequestCanceled(error)) {
  // handle the canceled request using cancelTokenSource.cancel();
}
```

RestApiConfigObject

Object containing the information required to invoke the custom REST API.

Properties

Name	Type	Description	Required
method	string	The request method to be used. Example: `GET, POST, PUT, PATCH, DELETE` **NOTE:** The default method is `GET`.	☐
headers	string	The custom headers to be sent along with the request.	☐
body	string	The data to be sent to the server as part of the request. **NOTE:** This is only applicable for the `PUT`, `POST`, and `PATCH` request methods.	☐
withoutDefaultHeaders	boolean	The flag that indicates whether default request headers must be sent along with the request. **NOTE:** • The default value is `false`.	☐

Name	Type	Description	Required
		• Set **withoutDefaultHeaders** to `true` if you do not want the default request headers to be sent along with the request. • Set **withoutDefaultHeaders** to `false` if you want the default request headers to be sent along with the request.	

RestAPIPayload

Object containing the properties that are sent along with a request to invoke a Pega REST API.

Properties

Name	Type	Description	Required
body	object	The data to be sent to the server. Example: `{ message: 'Hello World!', context: 'DATA-PORTAL $SpaceTra' }`	☐
queryPayload	object	The data of the query parameters to be used to prepare the URL of the REST API.	☐

Name	Type	Description	Required
		Example: `{ context: 'context', messageID: 'PEGASOCIAL M-56001' }`	
cancelTokenSource	object	The cancel token source object generated from the getCancelTokenSource() API.	☐
headers	object	The extra request headers to be sent along with the request.	☐
method	object	The request method to be sent to the server.	☐
responseType	object	The type of the response expected from the REST API.	☐
signal	object	The instance of the AbortSignal object that is used to communicate with or abort an asynchronous operation. This instance is returned by the AbortController interface.	☐

APIs in the SemanticUrlUtils class

Use the APIs in the SemanticUrlUtils class to build semantic URLs.

- **getActions()**
- **getResolvedSemanticURL(routeKey, payload, params)**

getActions()

Obtains actions supported by semantic URL utilities.

Returns

An object containing the supported actions.

Parameters

This API does not have parameters.

Usage example

In this example, the API returns an object containing the actions supported by semantic URL utilities.

```
const semanticUrlUtils = PCore.getSemanticUrlUtils();
const { ACTION_OPENWORKBYHANDLE } = semanticUrlUtils.getActions();
```

getResolvedSemanticURL(routeKey, payload, params)

Obtains the resolved semantic URL for a route key.

Returns

The resolved semantic URL as a string.

Parameters

Name	Type	Description	Required
routeKey	string	The type of action supported by the semantic URL utility.	☐
payload	object	The unique identifier that matches with app routes.	☐
params	object	The value assigned to the dynamic URL query parameter.	☐

Usage example

In this example, the API takes the supported action from the route key, identifies the app route from the payload, assigns the value to the query parameter, and returns the resolved semantic URL.

```
const semanticUrlUtils = PCore.getSemanticUrlUtils();
const routeKey = semanticUrlUtils.getActions().ACTION_OPENWORKBYHANDLE;
const payload = {caseClassName: "ON8TTL-MyApp-Work-MyCase"};
const params = {workID: "C-007"};
const resolvedURL = semanticUrlUtils.getResolvedSemanticURL(routeKey, payload, p
arams);
```

APIs in the StakeholderUtils class

Use the APIs in the StakeholderUtils class to handle the participants of a case.

- **createParticipant(caseID, participantRoleID, participantData, pConnectObj)**
- **deleteParticipant(caseID, participantID, pConnectObj)**
- **getParticipant(caseID, participantID, pConnectObj)**
- **getParticipantRoles(caseID, pConnectObj)**
- **getParticipants(caseID, pConnectObj)**
- **getRoleView(caseID, participantRoleID, pConnectObj)**
- **updateParticipant(caseID, participantID, participantData, pConnectObj)**

createParticipant(caseID, participantRoleID, participantData, pConnectObj)

Creates a new participant for a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case to which the new participant must be linked. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
participantRoleID	string	The string containing the new participant data.	☐
participantData	object	The data object containing the details of the participant.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API creates a new participant `Sam Smith` with role `Interested` for a case whose ID is `W-102`.

```
const participantData = {
 "content":{
  "pyFirstName":"Sam",
  "pyLastName":"Smith",
  "pyEmail1":"samsmith@test.com",
```

```
    "pyPhoneNumber":"+11234567899",
    "pyTitle":"Developer"
    }
  };
 PCore.getStakeholderUtils().createParticipant('ORG-MYAPP-WORK W-102','Interested
', participantData, getPConnect())
.then(newParticipantData => {
   // newParticipantData
}).catch(err => {
  // errors
});
```

deleteParticipant(caseID, participantID, pConnectObj)

Deletes an existing participant linked to a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose linked participant must be deleted. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
participantID	string	The ID of the participant to be deleted.	☐

Name	Type	Description	Required
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API deletes an existing participant with ID `Interested_02` linked to a case whose ID is `W-102`.

```
PCore.getStakeholderUtils().deleteParticipant('ORG-MYAPP-WORK W-102', 'Interested
_02', getPConnect())
.then(updatedParticipantData => {
   // updatedParticipantData
}).catch(err => {
  // errors
});
```

getParticipant(caseID, participantID, pConnectObj)

Obtains the data of a participant linked to a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case that the participant is linked to.	☐

Name	Type	Description	Required
		NOTE: Ensure that you provide the `pzInsKey` value of the **caseID**.	
participantID	string	The ID of the participant whose data must be obtained.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the data of the participant with participant ID `Interested_02`, linked to a case whose ID is `W-102`.

```
PCore.getStakeholderUtils().getParticipant('ORG-MYAPP-WORK W-102','Interested_02'
, getPConnect())
.then(participantData => {
// participantData
}).catch(err => {
// errors
});
```

getParticipantRoles(caseID, pConnectObj)

Obtains the list of participant roles for a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose list of participant roles must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the list of participant roles of a case whose ID is `W-102`.

```
PCore.getStakeholderUtils().getParticipantRoles('ORG-MYAPP-WORK W-102', getPConnect())
.then(roles => {
  // roles array
}).catch(err => {
  // errors
});
```

getParticipants(caseID, pConnectObj)

Obtains the list of participants for a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose list of participants must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the list of participants of a case whose ID is `W-102`.

```
PCore.getStakeholderUtils().getParticipants('ORG-MYAPP-WORK W-102', getPConnect())
.then(participants => {
  // participants array
}).catch(err => {
  // errors
});
```

getRoleView(caseID, participantRoleID, pConnectObj)

Obtains the view for a participant role for a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose participant role must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
participantRoleID	string	The ID of participant role whose view must be obtained.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the view of a participant whose role is `Owner` and is associated with a case whose ID is `W-102`.

```
PCore.getStakeholderUtils().getRoleView('ORG-MYAPP-WORK W-102', 'Owner', getPC
onnect())
.then(view => {
  // role view
```

```
}).catch(err => {
  // errors
});
```

updateParticipant(caseID, participantID, participantData, pConnectObj)

Updates the details of a participant for a case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose participant details must be updated. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
participantID	string	The ID of the participant whose details must be updated.	☐
participantData	object	The new participant data.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API updates the details of an existing participant with ID `Interested_02` for a case whose ID is `W-102`.

```
const updatedParticipantData = {
 "content": {
   "pyFirstName":"Sam",
   "pyLastName":"Smith"
 }
};
PCore.getStakeholderUtils().updateParticipant('ORG-MYAPP-WORK W-102','Interested
_02', updatedParticipantData, getPConnect())
.then(updatedParticipantData => {
   // updatedParticipantData
}).catch(err => {
  // errors
});
```

APIs in the StateUtils class

Use the APIs in the StateUtils class to perform actions related to the state of the Store.

- **getSharedState(key)**
- **getSuggestionsContext(context)**
- **setSharedState(key, value)**
- **updateState(context, key, value, options)**

getSharedState(key)

Obtains the property value from the shared state of the Store.

Returns

The property value from the shared state of the Store.

Parameters

Name	Type	Description	Required
key	string	The unique identifier of the property whose value must be obtained.	☐

Usage example

In this example, the value of the `sharedObject` property retrieved from the shared state of the Store is stored in the `callInfo` variable.

```
const callInfo = PCore.getStateUtils().getSharedState('sharedObject');
```

getSuggestionsContext(context)

Obtains an entry point to the SuggestionsContext object that contains APIs to perform actions on the Suggestions Context.

To view the APIs in the SuggestionsContext class, see APIs in the SuggestionsContext class.

Returns

The SuggestionsContext object.

Parameters

Name	Type	Description	Required
context	string	The name of the context based on	☐

Name	Type	Description	Required
		which the Suggestions Context is created.	

Usage example

In this example, the API returns the SuggestionsContext object for the `app/primary_1/workarea_1` context.

```
PCore.getStateUtils().getSuggestionsContext('app/primary_1/workarea_1');
```

setSharedState(key, value)

Assigns a value to a property in the shared state of the Store.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
key	string	The unique identifier of the property that must be assigned a value.	☐
value	any	The value that must be assigned to the property in the Store.	☐

Usage example

In this example, the API assigns a value to the `sharedObject` property.

```
PCore.getStateUtils().setSharedState('sharedObject', {
    number: '+(603) 345 456',
    status: 'Online'
});
```

updateState(context, key, value, options)

Updates the value of a property in the Store with the specified context and page reference.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
context	string	The context of the property that must be updated.	☐
key	string	The ID of the property whose value must be updated.	☐
value	any	The value that must be assigned to the property.	☐
options	object	The 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
pageReference	string	The reference to the page that contains the property that must be updated. **NOTE:** If **pageReference** is not mentioned, provide the complete path of the property in the key.	☐
isArrayDeepMerge	boolean	The flag that decides if the values within an array must be deep merged or replaced in the Store. **NOTE:** • The default value is `true`. • Set **isArrayDeepMerge** to `true` to deep merge the values within an array in the Store. • Set **isArrayDeepMerge** to `false` to replace the values within an array in the Store.	☐

Usage example

In this example, the API updates the value of the `CountryList` key to an empty array in the `app/modal_1` context by setting `isArrayDeepMerge` to false.

```
PCore.getStateUtils().updateState("app/modal_1", "CountryList", [], { pageReference: "caseInfo.content", isArrayDeepMerge: false });
```

APIs in the SuggestionsContext class

Use the APIs in the SuggestionsContext class to perform actions on the Suggestions Context.

- **getField(property)**
- **removeField(property)**
- **setField(property, value)**
- **setState(stateObj)**

getField(property)

Obtains the value of a specified field from the Suggestions Context.

Returns

The value of the field as a string.

Parameters

Name	Type	Description	Required
property	string	The field whose value is obtained from the Suggestions Context.	☐

Usage example

In this example, the API obtains the value of the `caseInfo.content.EmailField` field from the Suggestions Context.

```
PCore.getStateUtils().getSuggestionsContext('app/primary_1/workarea_1').getField('caseInfo.content.EmailField');
```

removeField(property)

Deletes the value of a specified field from the Suggestions Context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
property	string	The field whose value is deleted from the Suggestions Context.	☐

Usage example

In this example, the value of the `caseInfo.content.EmailField` field is deleted from the Suggestions Context.

```
PCore.getStateUtils().getSuggestionsContext('app/primary_1/workarea_1').removeField('caseInfo.content.EmailField');
```

setField(property, value)

Assigns a value to a specified field in the Suggestions Context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
property	string	The field that is assigned a value in the Suggestions Context.	☐
value	string	The value that is assigned to a field in the Suggestions Context.	☐

Usage example

In this example, the `suggestion@gmail.com` value is assigned to the `caseInfo.content.EmailField` field.

```
PCore.getStateUtils().getSuggestionsContext('app/primary_1/workarea_1').setField('caseInfo.content.EmailField','suggestion@gmail.com');
```

setState(stateObj)

Updates a state in the Suggestions Context.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
stateObj	object	The object containing the fields and values that must be updated in the Suggestions Context.	☐

Usage example

In this example, the API updates the values of the fields in the `caseInfo` object in the Suggestions Context.

```
PCore.getStateUtils().getSuggestionsContext('app/primary_1/workarea_1').setState({
    caseInfo: {
      content: {
        EmailField: 'suggestion@gmail.com',
        TextField: 'suggestion'
      }
    }
  })
```

APIs in the TagUtils class

Use the API in the TagUtils class to handle the tags of a case.

- **getTaggedCases(caseID, pConnectObj)**
- **getTags(caseID, pConnectObj)**
- **postTags(caseID, tags, pConnectObj)**

- **removeTag(caseID, tagID, pConnectObj)**

getTaggedCases(caseID, pConnectObj)

Obtains the view to render the list of cases associated with a specific tag.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case from where the view must be downloaded. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the view that renders the list of cases associated with a specific tag in a case whose ID is `W-102`.

```
PCore.getTagUtils().getTaggedCases('ORG-MYAPP-WORK W-102', getPConnect())
.then(tags => {
  // tags array
}).catch(err => {
```

```
  // errors
});
```

getTags(caseID, pConnectObj)

Obtains the tags of a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case whose tags must be obtained. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API obtains the tags of a case whose ID is `W-102`.

```
PCore.getTagUtils().getTags('ORG-MYAPP-WORK W-102', getPConnect())
.then(tags => {
  // tags array
}).catch(err => {
```

```
  // errors
});
```

postTags(caseID, tags, pConnectObj)

Adds tags to a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case to which the tags must be added. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
tags	array.<object>	The list of tags that must be added to the case.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API adds the tags, `Tag1` and `Tag2` to the case whose ID is `W-102`.

```
PCore.getTagUtils().postTags('ORG-MYAPP-WORK W-102', [{ Name : 'Tag1'}, {Name : 'Tag2'}], getPConnect())
.then(() => {
  // success
}).catch(err => {
   // Error handling
});
```

removeTag(caseID, tagID, pConnectObj)

Removes a tag from a specific case.

Returns

A Promise that resolves to an object.

Parameters

Name	Type	Description	Required
caseID	string	The ID of the case from which the tag must be removed. **NOTE:** Ensure that you provide the `pzInsKey` value of the **caseID**.	☐
tagID	string	The ID of the tag that is to be removed from the case.	☐
pConnectObj	object	The PConnect object of the component from where the API is being called.	☐

Usage example

In this example, the API removes the tag whose ID is `Tag1` from the case whose ID is `W-102`.

```
PCore.getTagUtils().removeTag('ORG-MYAPP-WORK W-102', 'Tag1', getPConnect())
.then(() => {
  // success
}).catch(err => {
  // errors
});
```

APIs in the UserApi class

Use the API in the UserApi class to handle user data.

- **getOperatorDetails(userID, isBusinessID)**

getOperatorDetails(userID, isBusinessID)

Obtains the glimpse data of a specified user.

Returns

The glimpse data as a Promise.

Parameters

Name	Type	Description	Required
userID	string	The ID of the user whose glimpse data must be obtained.	☐

Name	Type	Description	Required
isBusinessID	boolean	**NOTE:** This parameter is for internal use only. Please do not change its value.	☐

Usage example

In this example, the API obtains the glimpse data of a user whose ID is `author@constellation.com`.

```
PCore.getUserApi().getOperatorDetails('author@constellation.com')
  .then(response => {
      // The response of this API is as shown below:
      "data": [{
        "@class": "User",
        "ID": "Z2xvYmFsVXNlcl82M2RiZWQ2MGJlZGEzZTg2OTgyNTBiNGM",
        "Name": “Author Constellation”,
        "IsActive": true,
        "BusinessID": "author@constellation.com",
        "Email": "author@constellation.com",
        "AccessGroup": "globalGomechanicdefaultag"
      }],
      "pageNumber": 1,
      "pageSize": 1,
      "fetchDateTime": "2023-11-13T17:28:36.185Z")
    .catch(error => {
      console.log(error);
    });
```

APIs in the ViewResources class

Use the APIs in the ViewResources class to manage the view metadata in the rule store.

- **fetchViewResources(viewName, context, classID)**
- **updateViewResources(dxAPIResponse)**

fetchViewResources(viewName, context, classID)

Obtains the view metadata from the rule store.

Returns

The view metadata as an object.

Parameters

Name	Type	Description	Required
viewName	string	The name of the view rule.	☐
context	object	The context object. Example: `getPConnect()`	☐
classID	string	The class name of the case type.	☐

Usage example

In this example, the API returns the view metadata from the rule store.

```
const viewMetaData = PCore.fetchViewResources("viewname", getPConnect(), "OPB1HW-MyApp-Work-MyCase");
```

updateViewResources(dxAPIResponse)

Updates the view metadata in the rule store and loads the components.

Returns

Not applicable.

Parameters

Name	Type	Description	Required
dxAPIResponse	object	An object containing view metadata.	☐

Usage example

In this example, the API updates the view metadata in the rule store and loads the components.

```
PCore.updateViewResources(dxAPIResponse);
```

List of public constants

Use these publicly available constants through the *getConstants* API to access categorical information related to the Constellation architecture UI.

For more information about the *getConstants* API, see getConstants().

Application constants

Name	Description
`APP.APP`	Obtains the name of the app context.
`APP.PXREQUESTOR`	Obtains the name of the requestor page.

Name	Description
`APP.ROOT`	Obtains the name of the root context.

Case information constants

Name	Description
`CASE_INFO.ACTION_BUTTONS`	Obtains information about the action buttons in the current assignment.
`CASE_INFO.ASSIGNMENT_ID`	Obtains the ID of the current assignment.
`CASE_INFO.ASSIGNMENT_LABEL`	Obtains the name of the current assignment.
`CASE_INFO.ASSIGNMENTACTION_ID`	Obtains the ID of the current assignment action.
`CASE_INFO.ASSIGNMENTS`	Obtains the array of assignments belonging to the current case.
`CASE_INFO.AVAILABLEACTIONS`	Obtains the array of available actions that can be performed on the current case.
`CASE_INFO.CASE_INFO`	Obtains the information of the entire case including its content.
`CASE_INFO.CASE_INFO_CLASSID`	Obtains the class name of the current case.
`CASE_INFO.CASE_INFO_CONTENT`	Obtains the content of the current case.
`CASE_INFO.CASE_INFO_ID`	Obtains the ID of the current case.
`CASE_INFO.CASE_INFO_NAME`	Obtains the name of the current case.

Name	Description
`CASE_INFO.CASE_TYPE_ID`	Obtains the case type ID of the current case.
`CASE_INFO.CASE_TYPE_NAME`	Obtains the name of the current case type.
`CASE_INFO.CHILD_ASSIGNMENTS`	Obtains the array of child cases belonging to the current case.
`CASE_INFO.DUPLICATECASEID`	Obtains the ID of the duplicate case of the current case.
`CASE_INFO.HEADERS`	Obtains the request headers required for the current case.
`CASE_INFO.INSTRUCTIONS`	Obtains the instructions for the current assignment.
`CASE_INFO.NAVIGATION`	Obtains the navigation information of the current case. This information is used to view the life cycle of the case.
`CASE_INFO.PARENTCASEINFO`	Obtains the information of the parent case of the current case.
`CASE_INFO.REMOTESYSTEMID`	Obtains the remote system ID of the current remote case.
`CASE_INFO.STAGEID`	Obtains the current stage ID of the case.
`CASE_INFO.STAGES`	Obtains the information of the stages of the current case.
`CASE_INFO.VIEW_NAME`	Obtains the view name of the current assignment.

Local action constants

Name	Description
`LOCAL_ACTION_TYPE.CASE_WIDE`	Obtains the local action of type `case`.
`LOCAL_ACTION_TYPE.EXPRESS`	Obtains the local action of type `express`.

Message constants

Name	Description
`MESSAGES.MESSAGES_TYPE_ERROR`	Obtains the message type for an error message.
`MESSAGES.MESSAGES_TYPE_INFO`	Obtains the message type for an informational message.
`MESSAGES.MESSAGES_TYPE_SUCCESS`	Obtains the message type for a success message.

Page constants

Name	Description
`PAGE_TYPES.LANDINGPAGE`	Obtains the page type for a landing page.
`PAGE_TYPES.LISTPAGE`	Obtains the page type for a list page.
`PAGE_TYPES.PAGE`	Obtains the page type for a page.

Process constants

Name	Description
`PROCESS.FLOWPROBLEMS`	Obtains the name of the process for a problem flow.

Pub Sub Event constants

Name	Description
`PUB_SUB_EVENTS.EVENT_BULKACTION`	Obtains the event that notifies when a bulk action is submitted.
`PUB_SUB_EVENTS.EVENT_CANCEL`	Obtains the event that notifies when a cancel action is performed on an assignment.
`PUB_SUB_EVENTS.EVENT_EXPRESS_LOCALACTION`	Obtains the event that notifies when a local action of type express is submitted.

Name	Description
`PUB_SUB_EVENTS.EVENT_FULL_REAUTH`	Obtains the event that notifies complete reauthentication when both the access token and refresh token have expired for the Proof Key for Code Exchange (PKCE) flow.
`PUB_SUB_EVENTS.EVENT_REAUTH`	Obtains the event that notifies reauthentication.
`PUB_SUB_EVENTS.EVENT_CUSTOM_REAUTH`	Obtains the event that notifies reauthentication for a custom authentication flow.

Resource constants

Name	Description
`RESOURCE_STATUS.CREATE`	Obtains the status for the create operation performed on a resource.
`RESOURCE_STATUS.UPDATE`	Obtains the status for the update operation performed on a resource.
`RESOURCE_TYPES.ASSIGNMENT`	Obtains the resource type for assignment.
`RESOURCE_TYPES.CASE`	Obtains the resource type for case.
`RESOURCE_TYPES.DATA`	Obtains the resource type for data.
`RESOURCE_TYPES.PAGE`	Obtains the resource type for page.

View constants

Name	Description
`VIEW_NAMES.DATA_OBJECT_CREATE_VIEW`	Obtains the name of the default view for creating a data object.
`VIEW_NAMES.DATA_OBJECT_EDIT_VIEW`	Obtains the name of the default view for updating a data object.

Work basket constants

Name	Description
`WORK_BASKET.DATA_PAGES.D__PY_GET_USER_WORK_LIST_BY_USER_ID`	Obtains the name of the data page that contains the assignment list of a specific user.
`WORK_BASKET.DATA_PAGES.D__PY_MY_WORK_LIST`	Obtains the name of the data page that contains the assignment list of the current user.
`WORK_BASKET.DATA_PAGES.D__WORK_BASKET`	Obtains the name of the data page that contains the assignment list items of a specific user.

Miscellaneous constants

Name	Description
`BANNER_VARIANT_INFO`	Obtains the banner variant for an informational message.
`BANNER_VARIANT_SUCCESS`	Obtains the banner variant for a success message.
`BANNER_VARIANT_URGENT`	Obtains the banner variant for an urgent message.
`BANNER_VARIANT_WARNING`	Obtains the banner variant for a warning message.
`CREATE_DETAILS_VIEW_NAME`	Obtains the name of the view for the create stage.
`MODAL`	Obtains the name of the default modal view container.
`NEXT_ASSIGNMENT_INFO_ID`	Obtains the path to the ID of the next assignment.
`PREVIEW_VIEW_NAME`	Obtains the name of the view that displays a preview of the case.

List of OOTB events

Get notified when an out-of-the-box event occurs by subscribing to the event using the *subscribe* API from the PubSubUtils class..

For more information about the *subscribe* API, see subscribe(eventType, subscriptionItem, subscriptionItemName, subscribeOnce, contextName).

Name	Description	Payload properties	Sample payload
`constants.PUB_SUB_EVENTS.CASE_EVENTS.ASSIGNMENT_SUBMISSION`	This event is published when an assignment is submitted.	`caseID`, `isModalAction`, `isModalLaunchedFromPrimaryValue`, `isCaseWideAction`	{ "caseID": "OZ9ETS-CRED-WORK R-184334", "isModalAction": true, "isModalLaunchedFromPrimaryValue": false, "isCaseWideAction": false }
`constants.PUB_SUB_EVENTS.CASE_EVENTS.END_OF_ASSIGNMENT_PROCESSING`	This event is published when the user's assignments are completed.	`containerItemID`, `isCaseWideAction`, `caseID`, `assignmentID`, `actionID`	{ "isCaseWideAction": false, "caseID": "OZ9ETS-CRED-WORK R-184334", "assignmentID": "ASSIGN-WORKLIST OZ9ETS-CRED-WORK R-184334!PROCESS_FLOW", "actionID": null, "containerItemID": "app/primary_3/workarea_1

Name	Description	Payload properties	Sample payload
			" }

`constants.PUB_SUB_EVENTS.CASE_EVENTS.CASE_OPENED`	This event is published when a case is opened in the review mode using the openWorkByHandle(workID, className, options) API.	`caseKey`, `actionInContext`	{
			"caseKey": "OZ9ETS-CRED-WORK R-123342",
			"actionInContext": "app/primary_4"
			}

`constants.PUB_SUB_EVENTS.CASE_EVENTS.ASSIGNMENT_OPENED`	This event is published when an assignment is opened using PCore and PConnect APIs.	`actionInContext`, `assignmentID`, `caseKey`	{
			"actionInContext": "app/primary_5/workarea_1",
			"assignmentID": "ASSIGN-WORKLIST OZ9ETS-CRED-WORK R-123342!FEEDBACK_FLOW",
			"caseKey": "OZ9ETS-CRED-WORK R-123342"
			}

`constants.PUB_SUB_EVENTS.CASE_EVENTS.CASE_PREVIEW`	This event is published when the case is previewed using the	`caseId`, `context`	{ "caseId": "OZ9ETS-CRED-WORK%20R-123342", "context": "app/previe

Name	Description	Payload properties	Sample payload
	showCasePreview(pzInsKey, configObj) API.		w_1" }

`constants.PUB_SUB_EVENTS.EVENT_RENDER_APP`	This event is published when the application infrastructure has established the Redux store and is ready to perform its initial render.	`props`, `domContainerID`, `componentName`	{
			"props": {},
			"domContainerID": "app-root",
			"componentName": "RootContainer"
			}

`constants.PUB_SUB_EVENTS.EVENT_SHOW_CANCEL_ALERT`	This event is published when the user clicks cancel in
	• the create stage of a case
	• the create or update operations of a data object
		`isModalAction`, `isDataObject`, `hideDelete`	{
			isModalAction: true,
			hideDelete: false,
			isDataObject: undefined
			}

`constants.PUB_SUB_EVENTS.CASE_EVENTS.C`	This event is published when the user deletes	`caseType`	{ caseType: 'OZ9ETS-Cre

Name	Description	Payload properties	Sample payload
`REATE_STAGE_DELETED`	a case in the create stage.		d-Work-RedeemRewards' }
`constants.PUB_SUB_EVENTS.CASE_EVENTS.CREATE_STAGE_DONE`	This event is published when assignments in the create stage are completed by the user.	`caseId`, `assignmentId`, `caseType`	{ "caseId": "OZ9ETS-CRED-WORK R-184334", "assignmentId": "ASSIGN-WORKLIST OZ9ETS-CRED-WORK R-184334!PROCESS_FLOW", "caseType": "OZ9ETS-Cred-Work-Random" }
`constants.PUB_SUB_EVENTS.EVENT_EXPRESS_LOCALACTION`	This event is published when the express local action is submitted.	`submitResponse`	{ "submitResponse": { "data": { "caseInfo": { "content": { ... }, "ID": "OZ9ETS-CRED-WORK R-73336" ... } }, "uiResources": { ... }

Name	Description	Payload properties	Sample payload
			} }

`constants.PUB_SUB_EVENTS.EVENT_BULKACTION`	This event is published when the bulk action is submitted.	`submitResponse`	{
			"submitResponse": [
			{
			"response": {
			"results": [
			{
			"ID": "OZ9ETS-CRED-WORK R-73340",
			"BusinessID": "R-73340",
			"Name": "Redeem Rewards",
			"status": "200"
			}
			],
			"failureCount": "0"
			},
			"actionInPayload": {
			...
			}
			},
			{
			...
			}

Name	Description	Payload properties	Sample payload

			``
			  ]
			}
			``