Tokyo API Reference

Last updated: July 12, 2023

Company Headquarters
2225 Lawson Lane
Santa Clara, CA 95054
United States
(408)501-8550

Client API reference

Use client-side JavaScript APIs to control aspects of how ServiceNow is displayed and functions within the web browser. This reference lists available classes and methods along with parameters, descriptions, and examples to help control the end-user experience.

- api - UI Builder

  The api API provides methods that you can use when developing client scripts in the UI Builder.
- CustomEvent - Client

  You can use CustomEvent API to show qualified embedded help in the right sidebar.
- DynamicTranslation - Client

  The DynamicTranslation API provides methods that translate text, in real time, into multiple languages using translation service providers. This API is available for both standard clients and Angular-based Service Portal clients.
- g_service_catalog - Client

  The g_service_catalog API enables you to access data in a multi-row variable set (MRVS) when a model is open.
- GlideAgentWorkspace (g_aw) - Client

  The g_aw API enables a UI Action or client script to open a specified record in an Agent Workspace tab.
- GlideAjax - Client

  The GlideAjax class enables a client script to call server-side code in a script include.
- GlideDialogWindow - Client

  The GlideDialogWindow API provides methods for displaying a dialog in the current window and frame.

- GlideDocument - Client

  The GlideDocument class provides the ability to search a DOM element, a document, or a JQuery element.

- GlideFlow - Client

  Use the GlideFlow JavaScript API for client-side interactions with actions, flows, and subflows.

- GlideForm - Client

  The GlideForm API provides methods to customize forms.

- Mobile GlideForm (g_form) - Client

  Mobile GlideForm (g_form) methods enable you to work with forms on the mobile platform.

- GlideGuid - Client

  You can create a globally unique identifier.

- GlideList (Now Experience) - Client

  Use the GlideList API to customize lists in the Next Experience UI Framework.

- GlideList2 (g_list) - Client

  GlideList2 is a JavaScript class used to customize (v2) lists.

- GlideListV3 (g_list) - Client

  Use GlideListV3 to manipulate lists.

- GlideMenu (g_menu and g_item) - Client

  GlideMenu methods are used in UI Context Menus, in the onShow scripts to customize UI Context Menu items.

- GlideModalForm - Client

  Displays a form in a GlideModal.

- GlideModal - Client

  Provides methods for displaying a content overlay.

- GlideNavigation - Client

  Provides methods to control and refresh the navigator and main frame.

- GlideNotification - Client

  You can show messages over the page content.

- GlideRecord - Client

  GlideRecord is used for database operations. The client-side GlideRecord API enables the use of some GlideRecord functionality in client-side scripts, such as client scripts and UI policy scripts.

- GlideURLV3 - Client

  Provides methods for manipulating a URI.

- GlideUser - Client

  The GlideUser API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to get user information.

- GlideUIScripts - Client

  Access UI scripts from within client-side code.

- Guided Tours - Client

  Provides methods for launching and stopping guided tours.

- helpers - UI Builder

  The helpers API provides general functionality that is common across page scripts, eliminating the need to write scripts for simple functionality such as opening and closing a modal.

- i18N - Client

  Provides methods to get and format translated messages.

- openFrameAPI - Client

  OpenFrame is an omni-present frame that communication partners can use to integrate their systems into the ServiceNow platform.

- NotifyClient - Client

  The NotifyClient API allows you use Notify telephony functionality, such as making and receiving calls, from a web browser.

- NotifyOnTaskClient - Client

  The NotifyOnTaskClient API provides methods for sending SMS messages or starting/managing a conference call for various telephony service providers, such as Zoom and WebEx.

- ScopedSessionDomain - Client

  The ScopedSessionDomain API is a script include that contains client-side methods that provide functionality related to the current session domain.

- ScriptLoader - Client

  Provides the ability to load scripts asynchronously.

- SNAnalytics - Client

  The SNAnalytics API provides methods to push custom analytics data (events, pages, and user properties) to the User Experience Analytics for Service Portal dashboard.

- spAriaUtil - Client

  Show messages on a screen reader.

- spContextManager - Client

  Make data from a Service Portal widget available to other applications and services in a Service Portal page. For example, pass widget data to Agent Chat when it opens in a Service Portal page.

- spModal - Client

Show alerts, prompts, and confirmation dialogs in Service Portal widgets. The SPModal class is available in Service Portal client scripts.

- spUtil - Client

  Utility methods to perform common functions in a Service Portal widget client script.

- StopWatch - Client

  Use a StopWatch object to measure the duration of operations.

api - UI Builder

The api API provides methods that you can use when developing client scripts in the UI Builder.

This API is exposed to client scripts, also known as page scripts. Client scripts are executed in response to something happening on a page, such as:

- User interaction events/actions, such as a button click.
- Lifecycle events, such as a data broker execution started.

These scripts do not have to return anything to the framework and can be written as an asynchronous function.
This API is also exposed to scripted property values. These scripts are executed whenever the framework-runtime needs to calculate a value, such as:

- Passing to a component property.
- Determining component visibility.
- Emitting an event with a payload.

These scripts cannot be written as an asynchronous function. They also cannot invoke side-effect methods on the api object, such as, api.emit(), api.setState(), and api.data.<data_resource_id>.*().

The api object contains both configuration dependent and configuration independent properties that you can access within the context of the associated page or component. You cannot directly modify the properties within this object. Modification can only be made through the available methods.

api - api.context.props.<page_property_name>

Page properties can be configured within UI Builder. The configuration values depend on the context in which the page is used.

Field

Name	Type	Description

<page_property_name>	Any	Available values are dependant on the client script implementation.
		
		To access these properties, use the following: `api.context.props.<page_property_name>`.
		
		For example:
		`// A record page with property table could be accessed with`
		`function isActivityStreamVisible({api}) {`
		`  return api.context.props.table === 'incident';`
		`}`

Name	Type	Description
		Note: These property values are read-only. Mutating nested object values from scripts is not supported.

api - api.context.session.<session_property>

Context session properties associated with the current user.

Available session properties

Name	Type	Description
isLoggedIn	Boolean	Flag that indicates whether the current user is logged in to the system. Possible values: • true: Current user is logged in to • false: Current user is not logged in.
properties.awaEnabled	String	The system property glide.awa.enabled that indicates whether the auto assignment for work items for Advanced Work Assignment (AWA) is enabled for the current user. Possible values:

Name	Type	Description
		• true: AWA is enabled for the user. • false: AWA is not enabled for the user. For additional information, see Components installed with Advanced Work Assignment.
properties.forgetMe.value	String	The property glide.ui.forgetme that indicates whether to remove the **Remember Me** check box from the login page to prevent login information from being cached. Possible values: • true: Remove the **Remember Me** check box. • false: Display the **Remember Me** check box. For additional information, see Remove remember me.
properties.sessionTimeLeft.value	String	The system property glide.ui.session_timeleft that determines the mount of time left

Name	Type	Description
	Number coerced to string	before the current session times out. Use this property to prompt the user to extent the current session before it times out. Unit: Minutes
properties.sessionTimeout.value	String Number coerced to string	The system property glide.ui.session_timeout that determines the initial session time out value. Unit: Minutes - Values greater than 1440 minutes are treated as one day. For additional information, see Session activity timeout
properties.trackingEnabled.value	String	The system property glide.uxbuilder.tracking.enabled that indicates whether to enable/disable web analytics library loading and instantiation for UI Builder based applications. Possible values: • true: Enabled for the user.

Name	Type	Description
		• false: Disabled for the user.
user.avatar	String	URL of the current user's avatar.
user.dateFormat	String	Default date format.
user.domain	String	Domain path for the current user.
user.firstName	String	First name of the current user.
user.fullName	String	First and last name of the current user.
user.initials	String	Initials of the current user.
user.language	String	Primary language spoken by the current user.
user.preferences	Array of objects	Name-value pairs that describe the user preferences. These user preferences are stored as records in the User Preference [sys_user_preference] table, and are updated each time the user changes their settings. For additional information, see User preferences.

Name	Type	Description
user.roles	Array	Comma-separated list of roles assigned to the current user.
user.sys_id	String	Sys_id of the user in the User [sys_user] table.
user.timeFormat	String	Default time format to use for the user.
user.timeZone	String	Time zone of the current user.
user.timeZoneOffset	String	Time zone offset of the current user.

api - api.data.<data_resource_id>.lifecycle.lastFetchSucceeded

Boolean flag that indicates whether the last fetch attempt for the specified data resource instance finished successfully.

If the value is true, the last fetch attempt for the data resource instance finished successfully; otherwise, false.

Field

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and

Name	Type	Description
		defined when you add the data resource to your page in UI Builder.

api - api.data.<data_resource_id>.addErrorMessage(Object payload)

Displays the specified error message at the top of the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the error message to display.
		`"payload": {`
		`  "message": "String"`
		`}`

payload.message	String	Error message to display.

Returns

Type	Description
None	

Example

```
api.data.gform.addErrorMessage({message: 'Error message'})
;
```

api -api.data.<data_resource_id>.addInfoMessage(Object payload)

Displays the specified informational message at the top of the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the informational message to display.
		`"payload": {`
		`  "message": "String"`
		`}`

payload.message	String	Informational message to display.

Returns

Type	Description
None	

Example

```
api.data.gform.addInfoMessage({message: 'Test message'});
```

api - api.data.<data_resource_id>.addOption(Object payload)

Adds an option to the specified choice type field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field value to update.
		`"payload": {`
		`  "choiceIndex": "String",`
		`  "choiceLabel": "String",`
		`  "choiceValue": "String",`
		`  "fieldName": "String"`
		`}`

payload.choiceIndex	String	Optional. Index into the choice list at which to insert the option. Default: End of the choice list.
payload.choiceLabel	String	Label of the option to add to the specified field.
payload.choiceValue	String	Value of the option to add to the specified field.

Name	Type	Description
payload.fieldName	String	Name of the choice type form field to add the specified option to.

Returns

Type	Description
None	

Example

```
api.data.gform.addOption({fieldName: 'priority', choiceLabel: 'Extremely High', choiceValue: '10'});
```

api - api.data.<data_resource_id>.addWarningMessage(Object payload)

Displays the specified warning message at the top of the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the warning message to display.
		`"payload": {`
		`  "message": "String"`
		`}`

Name	Type	Description
payload. message	String	Warning message to display.

Returns

Type	Description
None	

Example

```
api.data.gform.addWarningMessage({message: 'Test message'}
);
```

api - api.data.<data_resource_id>.clearMessage()

Removes all informational and error messages from the top of the current form.

Parameters

Name	Type	Description
data_reso urce_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

```
api.data.gform.clearMessage();
```

api - api.data.<data_resource_id>.clearOptions(Object payload)

Clears all options from the specified choice type field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the choice type field whose options are to be cleared.
		`"payload": {`
		`  "fieldName": "String"`
		`}`

payload.fieldName	String	Name of the choice type field whose options are to be cleared.

Returns

Type	Description
None	

Example

```
api.data.gform.clearOptions({fieldName: 'priority'});
```

api - api.data.<data_resource_id>.executeUiAction(Object payload)

Executes the specified UI action.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the UI action to execute.
		`"payload": {`
		`  "actionSysId": "String"`
		`}`

payload.actionSysId	String	Sys_id of the UI action to execute. Located in the UI Action [sys_ui_action] table.

Returns

Type	Description
None	

Example

```
api.data.gform.executeUiAction({actionSysId: '60615ff90f73
0010ac7de6f8c4767e9a'});
```

api - api.data.<data_resource_id>.execute(Object inputValues)

Triggers an execute operation on the specified data resource.

This method is only available if the data resource is one of the following types:

- Composite
- GraphQL
- REST
- Scriptlet
- Transform

> **Note:** This method is only exposed if the mutates_server_data field is set to `true` on the corresponding data resource (sys_ux_data_broker_* table) record. It is accessible under api.data.<data_resource_Id>.refresh().

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.
inputValues	Object	Object to pass to the specified data resource. This object must conform to the data resource's input parameters.

Returns

Type	Description
None	

Example

This code example shows a page script that is invoked when the **Submit** button on the page is clicked. The page is configured wit a Server Data Resource tat creates a new record.

```
function handler({api}) {
  if (api.state.movieYear === 2020) {
    // The data resource used in this case specifies two i
nput parameters: name and year
    api.data.create_movie_record.execute({
      name: api.state.movieName,
      year: api.state.movieYear
    });
  }
}
```

api - api.data.<data_resource_id>.hideFieldMessage(Object payload)

Hides the oldest message next to the specified field or clears all messages associated with the field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field message to hide.
		`"payload": {`
		`  "clearAll": Boolean,`
		`  "fieldName": "String"`
		`}`

payload.clearAll	Boolean	Optional. Flag that indicates whether to clear all messages associated with the specified form field. Valid values:

Name	Type	Description
		• true: Clear all messages associated with the specified field. • false: Do not clear all messages associated with the specified field. Default: false
payload.fieldName	String	Name of the form field for which to hide the oldest message or clear all associated messages.

Returns

Type	Description
None	

Example

```
api.data.gform.hideFieldMessage({fieldName: 'short_descrip
tion'});
```

api - api.data.<data_resource_id>.hideRelatedList(Object payload)

Hides the specified related list on the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

payload	Object	Object that describes the related list to hide.
		`"payload": {`
		`  "listTableName ": "String"`
		`}`

payload.listTableName	String	Name of the related list to hide. Located in the Related List [sys_ui_related_list] table. If the list to hide is through a relationship, provide the sys_id of the list instead of the name.

Returns

Type	Description
None	

Example

```
api.data.gform.hideRelatedList({listTableName:'incident.parent_incident'});
```

api - api.data.<data_resource_id>.hideRelatedLists()

Hides all related lists on the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

```
api.data.gform.hideRelatedLists();
```

api - api.data.<data_resource_id>.refresh()

Triggers a refresh operation for the specified non-mutating data resource instance.

Call this method if the underlying data being fetched by the data resource changes. A data resource is considered non-mutating if the mutates_server_data field on the record is set to false.

This method is asynchronous and emits an internal event to trigger the refresh of the specified data resource instance. The UI Builder allows you to trigger client scripts in response to data resource lifecycle events, such as DATA_FETCH_SUCCEEDED and DATA_FETCH_FAILED. For additional information on these events, see Event mapping.

This method is only available if the data resource is one of the following types:

- Composite
- GraphQL
- REST
- Scriptlet
- Transform

Note: This method is only exposed if the mutates_server_data field is set to `false` on the corresponding data resource (sys_ux_data_broker_* table) record.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

This code example shows a page script that is invoked when a dropdown item is selected in a page. The page is configured with two Server Data Resources that query problem and incident tables.

```
function handler({api, event}) {
  const value = event.payload.value[0];
  if (value === 'problem')
    api.data.problem_list_1.refresh();
  else if(value === 'incident')
    api.data.incident_list_1.refresh();
}
```

api - api.data.<data_resource_id>.reload()

Reloads the current form using the same table and sys_id.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

```
api.data.gform.reload();
```

api - api.data.<data_resource_id>.removeOption(Object payload)

Removes an option from the specified choice type field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

Name	Type	Description

payload	Object	Object that describes the choice type field to update.
		`"payload": {`
		`  "choiceValue": "String",`
		`  "fieldName": "String"`
		`}`

payload.choiceValue	String	Value of the option to remove from the specified choice type field.
payload.fieldName	String	Name of the choice type form field from which to remove the specified value.

Returns

Type	Description
None	

Example

```
api.data.gform.removeOption({fieldName: 'priority', choice
Value: '1'});
```

api - api.data.<data_resource_id>.save()

Triggers form submission using the Save UI action.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

```
api.data.gform.save();
```

api - api.data.<data_resource_id>.setMandatory(Object payload)

Sets whether the specified form field is mandatory.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field whose mandatory information to update.
		`"payload": {`
		`  "fieldName": "String",`
		`  "mandatory": Boolean`
		`}`

Name	Type	Description
payload.fieldName	String	Name of the form field whose mandatory value is to be set.
payload.mandatory	Boolean	Flag that indicates the specified form field is mandatory, meaning the form cannot be saved without this field containing a valid value. Valid values: • true: Field is mandatory. • false: Field is optional.

Returns

Type	Description
None	

Example

```
api.data.gform.setMandatory({fieldName: 'short_description
', mandatory: false});
```

api - api.data.<data_resource_id>.setReadOnly(Object payload)

Sets the read/write capabilities of the specified form field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field whose readability information to update.
		`"payload": {`
		`  "fieldName": "String",`
		`  "readonly": Boolean`
		`}`

payload.fieldName	String	Name of the form field whose readability is to be set.
payload.readonly	Boolean	Flag that indicates the read/write capabilities of the specified form field. Valid values: • true: Field is read only. • false: Field is read/write.

Returns

Type	Description
None	

Example

```
api.data.gform.setReadOnly({fieldName: 'short_description'
, readonly: false});
```

api -api.data.<data_resource_id>.setValue(Object payload)

Updates a specified GlideForm field with the specified value. Optionally, you can also update the display value with the same specified value.

Only the value on the form is updated. The value is not saved in the database.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field whose value to update.
		`"payload": {`
		`  "displayValue": "String",`
		`  "fieldName": "String",`
		`  "value": "String"`
		`}`

payload.displayValue	String	Optional. Name of the display value to update. If left blank, the display value is not modified.
payload.fieldName	String	Name of the form field to update.
payload.value	String	Value to update the field with.

Returns

Type	Description
None	

Example

```
api.data.gform.setValue({fieldName: 'short_description', v
alue: 'short description'});
```

api - api.data.<data_resource_id>.setVisible(Object payload)

Sets the visibility of the specified form field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field on which to set visibility.
		`"payload": {`
		`  "fieldName": "String",`
		`  "visibility": Boolean`
		`}`

payload.fieldName	String	Name of the form field whose visibility is to be set.
payload.visibility	Boolean	Flag that indicates whether the associated field is visible on the current form. Valid values: • true: Field will display on the form. • false: Field will not display on the form.

Returns

Type	Description
None	

Example

```
api.data.gform.setVisible({fieldName: 'short_description'
, visibility: false});
```

api - api.data.<data_resource_id>.showFieldMessage(Object payload)

Displays the specified message next to the specified field.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the field message to display.
		`"payload": {`
		`  "fieldName": "String",`
		`  "message": "String",`
		`  "type": "String"`

payload.fieldName	String	Name of the field next to which the message should appear.

Name	Type	Description
payload. message	String	Message to display.
payload.t ype	String	Optional. Type of message to display. Valid values: • error • info • warning Default: info

Returns

Type	Description
None	

Example

```
api.data.gform.showFieldMessage({fieldName: 'short_descrip
tion', message: 'Error message', type: 'error'});
```

api - api.data.<data_resource_id>.showRelatedList(Object payload)

Displays the specified related list on the current form.

Parameters

Name	Type	Description
data_reso urce_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

payload	Object	Object that describes the related list to display.
		`"payload": {`
		`  "listTableName ": "String"`
		`}`

payload.listTableName	String	Name of the related list to display. Located in the Related List [sys_ui_related_list] table. If the list to display is through a relationship, provide the sys_id of the list instead of the name.

Returns

Type	Description
None	

Example

```
api.data.gform.showRelatedList({listTableName:'incident.pa
rent_incident'});
```

api - api.data.<data_resource_id>.showRelatedLists()

Displays all related lists associated with the current form.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when

Name	Type	Description
		you add the data resource to your page in UI Builder.

Returns

Type	Description
None	

Example

```
api.data.gform.showRelatedLists();
```

api - api.data.<data_resource_id>.submit()

Triggers form submission using the specified UI action.

Parameters

Name	Type	Description
data_resource_id	String	Unique identifier of the associated data resource. The data resource for this method must be based off of Glide Form (gform.) The available data resource instances are configuration-dependent and defined when you add the data resource to your page in UI Builder.

payload	Object	Object that describes the UI action to use to submit the current form.
		`"payload": {`
		`  "submitActionName": "String"`
		`}`

Name	Type	Description
payload.submitActionName	String	Name of the UI action to execute to submit the current form.

Returns

Type	Description
None	

Example

```
api.data.gform.submit({submitActionName:'sysverb_ws_save'}
);
```

api - api.emit(String eventName, Object payload)

Emits an event with the specified name and payload.

The event name being emitted must be part of the associated page definition's dispatched events list, which is stored in the UX Macroponent Definition [sys_ux_macroponent] table. Any api.emit call that dispatches events not declared in this table are ignored.

For additional information on events, see Work with events.

Parameters

Name	Type	Description
eventName	String	Name of the event to emit. This name should follow the UI Framework action naming guidelines: • Should be upper snake case, such as ITEM_CHANGED. All letters in upper case and all spaces replaced with underscores. • Should be past tense, such as BUTTON_CLICKED or USER_SELECTED.

Name	Type	Description
		For additional information on these action naming guidelines, see https://developer.servicenow.com/dev.do#!/reference/now-experience/sandiego/ui-framework/main-concepts/dispatching-actions.
payload	Object	Optional. Object that contains the data to send with the emitted event. This object is free-form and can contain whatever data is necessary by the entity receiving the data. **Note:** Payloads of primitive type work, but could result in inconsistent behavior.

Returns

Type	Description
None	

Example

The following code example shows emitting an event called NOW_UXF_PAGE#ADD_NOTIFICATIONS with an associated items payload.

```
function handler({api}) {
  api.emit('NOW_UXF_PAGE#ADD_NOTIFICATIONS', {
    items: [
      {
        id: 'alert1',
        status: 'positive',
        icon: 'check-circle-outline',
        content: 'Here is some information!',
        textLinkProps: {
          label: 'More info',
          href: 'https://www.servicenow.com'
        },
      action: {type: 'acknowledge'}
```

```
        }
      ]
    });
}
```

api - setState(String stateParam, Any value)

Sets the value of the specified client state parameter.

Use client state parameters to maintain a shared state on a page. The shared state can then be passed as values to properties of components used on the page. You can also access and update client states in multiple page scripts. A common use case is to keep track of values input by users in multiple form controls on a page. When the form is submitted, a client script could then use all of the values stored in client state parameters to create a new record with a data broker. A page can have one or more client state parameters, which you can declare for a page through the UI Builder. You can then bind a client state parameter to one or more components to share or act on the client state parameter.

api.setState() calls are executed asynchronously and do not necessarily update the UI immediately. If the value to set depends on the currentValue of the client state parameter, or any of the properties provided in the api object, you should use this variant of the api.setState() method to avoid using outdated values.

Parameters

Name	Type	Description
stateParm	String	Name of the client state parameter to update. This name must be declared in the client state parameters of the associated page. For additional information on declaring client state parameters, see Work with client state parameters.
value	Any - Must be the same as	Value to set the specified client state parameter to.

Name	Type	Description
	the client state paramete r declaratio n.	

Returns

Type	Description
None	

Example

This example shows a script that could be executed to update the email client state parameter when an input value is set on a now-input component.

```
function handler({api, event}) {
    api.setState('email', event.payload.value);
}
```

api - setState(String stateParam, Function callbackFn)

Sets the value of the specified client state parameter to the value returned by the specified callback function.

The callback function is invoked with an object that has two properties: currentValue and api. The function should never mutate the currentValue property or the api object directly.

Use client state parameters to maintain a shared state on a page. The shared state can then be passed as values to properties of components used on the page. You can also access and update client states in multiple page scripts. A common use case is to keep track of values input by users in multiple form controls on a page. When the form is submitted, a client script could then use all of the values stored in client state parameters to create a new record with a data broker. A page can have one or more client state parameters, which you can declare for a page through the UI Builder. You can then bind a client state parameter to one or more components to share or act on the client state parameter.

api.setState() calls are executed asynchronously and do not necessarily update the UI immediately. If the value to set depends on the currentValue of the client state parameter, or any of the properties provided in the api object, you should use this variant of the api.setState() method to avoid using outdated values.

Parameters

Name	Type	Description
stateParm	String	Name of the client state parameter to update. This name must be declared in the client state parameters of the associated page. For additional information on declaring client state parameters, see Work with client state parameters.
callbackFn	Function	Callback function to execute to obtain the value.

Returns

Type	Description
None	

Example

This example shows how to use api.setState to log users into a page.

```
function handler({api, event}) {
  const {name, value} = event.payload;
  if (name === 'username' || name === 'password') {
    // Update the loginParameters state object with the us
ername/password value
    api.setState('loginParameters', ({currentValue}) => {
      return {
        ...currentValue,
        [name]: value
      };
    });
  }
}
```

api - api.state.<client_state_parameter_name>

Current value of the specified client state parameter.

Field

Name	Type	Description

<client_state_parameter_name>	Any. The available client state parameters are dependent on the page configuration.	Name of the client state parameter. Available client states are dependant on the client script implementation.
		
		To access the available client states, use the following: `api.state.<client_state_name>`.
		
		For example:
		`function showRelatedLists({api})`
		`{`
		`  return !api.state.isCustomListSelected;`
		`}`

Name	Type	Description
		Note: These property values are read-only. To update a client state parameter, use api.setState(). Mutating nested object values from scripts is not supported.

CustomEvent - Client

You can use CustomEvent API to show qualified embedded help in the right sidebar.

See Use embedded help qualifiers for more information.

CustomEvent - fireAll(String event, String qualifier)

Show the embedded-help content specified by the qualifier parameter in the right sidebar.

Before using the fireAll() method, you must have created the Embedded Help qualifier and help content.

Note: To write to the debug log, make a call to the global function `jslog()`.

Parameters

Name	Type	Description
event	String	The event to send. Must be the string "embedded_help:load_embedded_help"

Name	Type	Description
qualifier	String	The qualifier name created in the Embedded Help application.

Returns

Type	Description
None	

Example

```
var qualifier = 'your-EH-qualifier';
                CustomEvent.fireAll("embedded_help:load_em
bedded_help", qualifier);
```

DynamicTranslation - Client

The DynamicTranslation API provides methods that translate text, in real time, into multiple languages using translation service providers. This API is available for both standard clients and Angular-based Service Portal clients.

In addition, you can use this API to detect the language of a specific string and check whether the DynamicTranslation methods are enabled for a translation service. Use this API to create a seamless localization experience for your user interface, enabling one interface to service multiple countries.

Currently this API supports three translation service providers: Microsoft Azure Translator Service, IBM Watson Translator Service, and Google Cloud Translator Service. You can also configure other translation services within your instance and then use the DynamicTranslation API to translate your text.

To use this API you must activate the Dynamic Translation plugin. For information on this plugin and additional information on Dynamic Translation, refer to Dynamic translation overview. Also, to use this API in a

Service Portal widget, you must inject the dynamicTranslation service into the widget client script function.

> **Note:** The name of the class to use in Service Portal clients is dynamicTranslation, while the name of the class to use in standard clients is DynamicTranslation.

DynamicTranslation - getDetectedLanguage(String text, Object parms)

Detects the language of the passed in text.

If you pass in a translator, the method uses that translation service to detect the source language. Otherwise, the detection is performed by the default translation service. Ensure that the text strings that you provide contain enough verbiage to enable proper language detection.

In addition to the detected language, the response contains a confidence level of the detection, along with other possible language alternatives. If a translator is not passed in, the method also returns the default translation service used to detect the language.

Parameters

Name	Type	Description
text	String	Text to use to detect the language.
parms	Object	Optional. JSON object that contains additional translation parameters. `"parms": { "translator": " String" }`
parms.translator	String	Translation service to use to detect the language of a string.

Name	Type	Description
		Translation services are configured under the **Translator Configuration** menu and located in the Translator Configuration [sn_dt_translator_configuration] table. Possible values - not case-sensitive: • Google • Microsoft • IBM • <custom> **Note:** To use custom translation services you must first configure the translation service in your instance. For details, see Integrate with a translation service provider. Default: Translation service configured in the Translator Configuration [sn_dt_translator_configuration] table.

Returns

Type	Description
alternatives	Array of objects that describe other languages that might also be a match. Data type: Array `"alternatives": [ { "code": "String", "confidence": "String", "name": "String" } ]`
alternatives.code	Language code of the alternative language. Data type: String
alternatives.confidence	Float value indicating the confidence level of the alternative language. Value is between zero and one. The lower the value, the lower the confidence level. Data type: String
alternatives.name	Language code of the alternative language. Data type: String
detectedLanguage	Description of the detected language.

Type	Description

	Data type: Object
	`"detectedLanguage": {`
	`  "code": "String",`
	`  "confidence": "String",`
	`  "name": "String"`
	`}`

detectedLanguage.code	Language code of the detected language. Data type: String
detectedLanguage.confidence	Float value indicating the confidence level of the alternative language. Value is between zero and one. The lower the value, the lower the confidence level. Data type: String
detectedLanguage.name	Language code of the detected language. Data type: String
translator	Translation service used to detect the language. Data type: String
Error messages	The following are error messages that the method may return and indications as to the error's root cause. • Text ("text" field) is missing or invalid. (40000): The text to

Type	Description
	detect the language is either missing or not a string. • Dynamic Translation plugin is not installed. (40001): The Dynamic Translation API was invoked without activating the com.glide.dynamic_translation plugin. For information on activating this plugin, see Dynamic translation overview. • Translator ("translator" field) is invalid. (40003): The passed in translator parameter is not a string. • <translator> translator is not configured. (40004): The specified translation service is not configured in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is inactive. (40005): The specified translation service is not set to **Active** in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Additional parameters are invalid. (40006): The additional parameters that were passed are not an object. • Maximum time limit has been exceeded. (40009): The operation took longer than the

Type	Description
	defined timeout value specified in the Translation Configuration. Default: 40 seconds • Default translator is not configured for detection. (40011): The default translation service has not been specified for language detection in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is not configured for detection. (40013): The specified translation service is not configured for language detection in the Translator Configuration. For information on creating/ modifying a translator configuration, see Create a translator configuration. • Unknown error occurred. (40051): Default error thrown when the error doesn't fall in to any other category. • Text ("text" field) has exceeded its maximum length. (40052): The text that was passed in for language detections exceeds the maximum length supported by the corresponding translation service. • Request is not authorized because credentials are missing or invalid (40055): The credentials

Type	Description
	configured for the translation service in Connections & Credentials are not valid. For information on connections and credentials, see Dynamic translation overview.

Example

This example shows code that detects a string in English using IBM's translation service in a standard client script.

```
var detectedResponse = DynamicTranslation.getDetectedLangu
age('Please detect the language of this text', {"translato
r":'IBM'}).then(function(res) {console.log(res); }, functi
on(res) {console.log(res); } );
```

Output:

```
detectedResponse {
  detectedLanguage:
    { "code": "en", "confidence": "1", "name": "en" }
  alternatives:
    [
      { "code": "vi", "confidence": "0.86", "name": "vi" }
,
      { "code": "id", "confidence": "0.86", "name": "id" }
    ]
 }
```

Example

This example shows a client script that throws an error when an invalid translation service is passed in.

```
var detectedResponse = DynamicTranslation.getDetectedLangu
age('Please detect the language of this text', {"translato
r":123}).then(function(res) {console.log(res); }, function
(res) {console.log(res); } );
```

Output:

```
{"code":"40003","message":"Translator (\"translator\" fiel
d) is invalid"}
```

Example

This example shows code that detects a string in English using IBM's translation service in a Service Portal widget client script. Note that the name of the class is `dynamicTranslation` not `DynamicTranslation`.

```
var detectedResponse = dynamicTranslation.getDetectedLangu
age('Please detect the language of this text', {"translato
r":'IBM'}).then(function(res) {console.log(res); }, functi
on(res) {console.log(res); } );
```

Output:

```
detectedResponse {
  detectedLanguage:
    { "code": "en", "confidence": "1", "name": "en" }
  alternatives:
    [
      { "code": "vi", "confidence": "0.86", "name": "vi" }
,
      { "code": "id", "confidence": "0.86", "name": "id" }
    ]
 }
```

Example

This example shows a Service Portal widget client script that throws an error when an invalid translation service is passed in.

```
var detectedResponse = dynamicTranslation.getDetectedLangu
age('Please detect the language of this text', {"translato
r":123}).then(function(res) {console.log(res); }, function
(res) {console.log(res); } );
```

Output:

```
{"code":"40003","message":"Translator (\"translator\" fiel
d) is invalid"}
```

DynamicTranslation - getDetectedLanguages(Array texts, Object parms)

Detects the languages of the passed in text strings.

If you pass in a translator, the method uses that translation service to detect the source language. Otherwise, the detection is performed by the default translation service. Ensure that the text strings that you provide contain enough verbiage to enable proper language detection.

In addition to the detected language, the response contains a confidence level of the detection, along with other possible language alternatives. If a translator is not passed in, the method also returns the default translation service used to detect the language.

When calling this method from a portal client script, use the class name dynamicTranslation; such as dynamicTranslation.getTranslations(). When calling it from a platform client script, use the class name DynamicTranslation; such as DynamicTranslation.getTranslations().

Parameters

Name	Type	Description

parms	Object	Optional. JSON object that contains additional translation parameters.
		`"parms": {`
		`  "translator": "String"`
		`}`

parms.translator	String	Translation service to use to detect the language of a string. Translation services are configured under the **Translator Configuration** menu and located in the Translator Configuration [sn_dt_translator_configuration] table. Possible values - not case-sensitive: • Google • Microsoft

Name	Type	Description
		• IBM • <custom> **Note:** To use custom translation services you must first configure the translation service in your instance. For details, see Integrate with a translation service provider. Default: Translation service configured in the Translator Configuration [sn_dt_translator_configuration] table.
texts	Array	List of text strings to use to detect the language(s).

Returns

Type	Description

detections	Language detection of text strings.
	
	Data type: Object
	
	`"detections": {`
	`  "alternatives": [Array],`
	`  "detectedLanguage": {Object},`
	`  "isError": Boolean`
	`]`

detections.alternatives	Array of objects that describe other languages that might also be a match.
	
	Data type: Array
	
	`"alternatives": [`
	`  {`
	`    "code": "String",`

Type	Description
	```
	    "confidence": "String",
	    "name": "String"
	  }
	]
	```
detections.alternatives.code	Language code of the alternative language. Data type: String
detections.alternatives.confidence	Float value indicating the confidence level of the alternative language. Value is between zero and one. The lower the value, the lower the confidence level. Data type: String
detections.alternatives.name	Language code of the alternative language. Data type: String

detections.detectedLanguage	Description of the detected language.
	
	Data type: Object
	```
	"detectedLanguage": {
	  "code": "String",
	  "confidence": "String",
	  "name": "String"
	}
	```

detections.detectedLanguage.code	Language code of the detected language. Data type: String
detections.detectedLanguage.confidence	Float value indicating the confidence level of the alternative language. Value is between zero and

Type	Description
	one. The lower the value, the lower the confidence level. Data type: String
detections.detectedLanguage.name	Language code of the detected language. Data type: String
detections.isError	Flag that indicates whether the detection of language for the text resulted in an error. Valid values: • true: Error encountered. • false: Language detection was successful. Data type: Boolean
status	Status of the response to the method call. Possible values: • Error • Partial • Success Data type: String
translator	Translation service used to detect the language. Data type: String
Error messages	The following are error messages that the method may return and indications as to the error's root cause.

Type	Description
	• Text ("text" field) is missing or invalid. (40000): The text to detect the language is either missing or not a string. • Dynamic Translation plugin is not installed. (40001): The Dynamic Translation API was invoked without activating the com.glide.dynamic_translation plugin. For information on activating this plugin, see Dynamic translation overview. • Translator ("translator" field) is invalid. (40003): The passed in translator parameter is not a string. • <translator> translator is not configured. (40004): The specified translation service is not configured in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is inactive. (40005): The specified translation service is not set to **Active** in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Additional parameters are invalid. (40006): The additional parameters that were passed are not an object. • Maximum time limit has been exceeded. (40009): The operation took longer than the defined timeout value specified in the Translation Configuration. Default: 40 seconds • Request failed with multiple errors. (40010): Multiple errors occurred in the language detection call. For more information, refer to the response for each individual text string. • Default translator is not configured for detection. (40011): The default translation service has not been specified for language detection in the Translator Configuration. For information on

Type	Description
	creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is not configured for detection. (40013): The specified translation service is not configured for language detection in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Translator configuration version is invalid. Migrate to v3. (40014): The associated version of the Translator Configuration for the specified translation service does not support the specified text translation method. For more information, see Migrate to version v3 of a translator configuration. • Unknown error occurred. (40051): Default error thrown when the error doesn't fall in to any other category. • Text ("text" field) has exceeded its maximum length. (40052): The text that was passed in for language detections exceeds the maximum length supported by the corresponding translation service. • Request is not authorized because credentials are missing or invalid (40055): The credentials configured for the translation service in Connections & Credentials are not valid. For information on connections and credentials, see Dynamic translation overview.

Example

This example shows code from a portal client script that detects English as the language of the passed-in strings using the Microsoft translation service.

```
var detectedResponse = dynamicTranslation.getDetectedLangu
ages(["First text string language to detect", "Second tex
t string language to detect"], {"translator": "Microsoft"}
).then(function(res) {console.log(res);}, function(res) {c
onsole.log(res);});
gs.info(JSON.stringify(detectedResponse));
```

Output

```
{
  "translator":"Microsoft",
  "status":"Success",
  "detections":[
    {
      "isError":false,
      "detectedLanguage":{"name":"en", "code":"en", "confi
dence":"1"},
      "alternatives":[
        {"name":"id", "code":"id", "confidence":"0.83"},
        {"name":"ms", "code":"ms", "confidence":"0.83"}
      ]
    },
    {
      "isError":false,
      "detectedLanguage":{"name":"en", "code":"en", "confi
dence":"1"},
      "alternatives":[
        {"name":"fr", "code":"fr", "confidence":"0.83"},
        {"name":"id", "code":"id", "confidence":"0.83"}
      ]
    }
  ]
}
```

Example

This example shows code in a portal client script that returns a Partial status when two text strings are passed in and one of them is invalid. To use this code example in a platform client script, change `dynamicTranslation.getDetectedLanguages` to `DynamicTranslation.getDetectedLanguages`.

```
var detectedResponse = dynamicTranslation.getDetectedLangu
ages(["First text string language to detect", ""], {"trans
lator": "Microsoft"}).then(function(res) {console.log(res)
; }, function(res) {console.log(res); } );
gs.info(JSON.stringify(detectedResponse));
```

Output

```
{
  "translator":"Microsoft",
  "status":"Partial",
  "detections":[
    {
      "isError":false,
      "detectedLanguage":{"name":"en", "code":"en", "confi
dence":"1"},
      "alternatives":[
        {"name":"id", "code":"id", "confidence":"0.83"},
        {"name":"ms", "code":"ms", "confidence":"0.83"}
      ]
    },
    {
      "isError":true,
      "code":"40000",
      "message":"Text is missing or invalid"
    }
  ]
}
```

Example

This example shows code from a portal client script that throws an error when an invalid translation service is passed in. To use this code example for a platform client script, change `dynamicTranslation.getDetectedLanguages` to `DynamicTranslation.getDetectedLanguages`.

```
var detectedResponse = dynamicTranslation.getDetectedLangu
ages(["First text string language to detect", "Second tex
t string language to detect"], {"translator": "123"}).then
(function(res) {console.log(res); }, function(res) {consol
e.log(res); } );
gs.info(JSON.stringify(detectedResponse));
```

Output

```
{"code":"40003","message":"Translator (\"translator\" fiel
d) is invalid","status":"Error"}
```

DynamicTranslation - getTranslation(String textToTranslate, Object parms)

Translates the passed in text to one or more languages.

The method uses translation services, such as Microsoft Azure Translator Service, IBM Watson Translator Service, and Google Cloud Translator Service, to perform the translation. If you do not pass in translation parameters, the method uses the system default.

Parameters

Name	Type	Description
textToTranslate	String	Text to translate.
parms	Object	Optional. JSON object that contains additional translation parameters. `"parms": { "additionalParameters": {Object}, "sourceLanguage": "String", "targetLanguages": [Array], "translator": "String" }`
parms.additionalParameters	Object	Optional. Array of JSON objects. Each object contains key-value pairs that provide additional information for

Name	Type	Description
		performing the translation. `"additionalParameters": { "parameterName": "String", "parameterValue": "String" }`
parms.additionalParameters.parameterName	String	Optional. Key name. Valid values: textype: Type of text to translate. For Microsoft Azure Translator Service only.
parms.additionalParameters.parameterValue	String	Optional. Value of the associated key. Valid values: • plain: Standard text string • html: HTML text string Default: plain
parms.sourceLanguage	String	Optional. Language code of the source text. Default: Translation service detects the source language.

Name	Type	Description
parms.targetLanguages	Array	Optional. List of language codes to use to translate the text. The method returns translated text for each language code. Default: User preferred language.
parms.translator	String	Optional. Translation service to use to translate the text (not case-sensitive). Valid values: • Google • Microsoft • IBM • <custom> **Note:** To use custom translation services you must first configure the translation service in your instance. For details, see Integrate with a translation service provider. Default: Translation service configured in the Translator

Name	Type	Description
		Configuration [sn_dt_translator_configuration] table.

Returns

Type	Description

detectedLanguage	Description of the detected language.
	
	Data type: Object
	
	`"detectedLanguage": {`
	`  "code": "String",`
	`  "name": "String"`
	`}`

detectedLanguage.code	Language code of the detected language. Data type: String
detectedLanguage.name	Language code of the detected language. Data type: String

translations	Array of objects that describe the language translations.
	
	Data type: Array
	
	`translations": [`
	`  {`
	`    "targetLanguage": "String",`
	`    "translatedText": "Stri`

Type	Description

	`ng"`
	`   }`
	`]`

translations.targetLanguage	Language code to which the source text was translated. Data type: String
translations.translatedText	Translated text. Data type: String
translator	Translation service used to detect the language. Data type: String
Error messages	The following are error messages that the method may return and indications as to their root cause. • Text ("text" field) is missing or invalid. (40000): The text to translate is either missing or not a string. • Dynamic Translation plugin is not installed. (40001): The Dynamic Translation API was invoked without activating the com.glide.dynamic_translation plugin. For information on activating this plugin, see Dynamic translation overview. • Default translator is not configured for translation. (40002): No translation

Type	Description
	service is selected as the default translation service in the Translator Configurations. For information on creating/modifying a translator configuration, see Create a translator configuration. • Translator ("translator" field) is invalid. (40003): The passed in translator parameter is not a string. • <translator> translator is not configured. (40004): The specified translation service is not configured in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is inactive. (40005): The specified translation service is not set to **Active** in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Additional parameters are invalid. (40006): The additional parameters that were passed are not an object. • Target languages ("targetLanguages" field) are invalid. (40007): The targetLanguages parameter is

Type	Description
	passed in the call but is not valid for one of the following reasons: • Value is not an array • Array is empty • One or multiple of the entries is not a string • Source language ("sourceLanguage" field) is invalid. (40008): The sourceLanguage parameter is passed in the call but the value is not a String. • Maximum time limit has been exceeded. (40009): The operation took longer than the defined timeout value specified in the Translation Configuration. Default: 40 seconds • <translator> translator is not configured for translation. (40012): The specified translation service is not configured for text translation in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Unknown error occurred. (40051): Default error thrown when the error doesn't fall in to any other category. • Text ("text" field) has exceeded its maximum length. (40052): The text that was passed in to translate exceeds the

Type	Description
	maximum length supported by the corresponding translation service. • Source language is invalid. (40053): The passed in sourceLanguage parameter contains a language code that is not supported by the corresponding translation service. • Target language is invalid. (40054): One or more of the language codes passed in the targetLanguages parameter is not supported by the corresponding translation service. • Request is not authorized because credentials are missing or invalid (40055): The credentials configured for the translation service in Connections & Credentials are not valid. For information on connections and credentials, see Dynamic translation overview. • Text cannot be translated to target languages. (40056): The specified translation service is not able to translate the passed in text into the specified target languages.

Example

This example shows the translation of plain text content from English (detected) into French and Italian using Microsoft's translation service in a standard client script.

```
DynamicTranslation.getTranslation("Translate this text usi
ng platform from client", {
  "targetLanguages": ["fr", "it"],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```

{"translations":[
    {
      "targetLanguage":"it",
      "translatedText":"Tradurre questo testo utilizzando
la piattaforma dal client"
    },
    {
      "targetLanguage":"fr",
      "translatedText":"Traduire ce texte en utilisant la
plate-forme du client"
    }
  ],
  "translator":"Microsoft",
  "detectedLanguage":{"code":"en","name":"en"}
}
```

Example

This example shows a client script that throws an error when an invalid target language is passed in.

```
DynamicTranslation.getTranslation("Translate this text usi
ng platform from client", {
  "targetLanguages": ["123”],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```
{"code":"40054","message":"Target language is invalid"}
```

Example

This example shows the translation of plain text content from English (detected) into French and Italian using Microsoft's translation service in a Service Portal widget client script. Note that the name of the class is `dynamicTranslation` not `DynamicTranslation`.

```
dynamicTranslation.getTranslation("Translate this text usi
ng platform from client", {
  "targetLanguages": ["fr", "it"],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```

{"translations":[
    {
      "targetLanguage":"it",
      "translatedText":"Tradurre questo testo utilizzando
la piattaforma dal client"
    },
    {
      "targetLanguage":"fr",
      "translatedText":"Traduire ce texte en utilisant la
plate-forme du client"
    }
  ],
  "translator":"Microsoft",
  "detectedLanguage":{"code":"en","name":"en"}
}
```

Example

This example shows a Service Portal widget client script that throws an error when an invalid target language is passed in

```
dynamicTranslation.getTranslation("Translate this text usi
ng platform from client", {
  "targetLanguages": [123],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```
{"code":"40054","message":"Target language is invalid"}
```

DynamicTranslation - getTranslations(Array texts, Object parms)

Translates the passed in text strings to one or more languages.

The method uses translation services, such as Microsoft Azure Translator Service, IBM Watson Translator Service, and Google Cloud Translator Service, to perform the translation. If you do not pass in translation parameters, the method uses the system default.

When calling this method from a portal client script, use the class name dynamicTranslation; such as dynamicTranslation.getTranslations(). When calling it from a platform client script, use the class name DynamicTranslation; such as DynamicTranslation.getTranslations().

Parameters

Name	Type	Description
texts	Array	List of text stings to translate.

parms	Object	Optional. JSON object that contains additional translation parameters.
		`"parms": {`
		`  "additionalParameters": {Object},`
		`  "sourceLanguage": "String",`
		`  "targetLanguages": [Array],`
		`  "translator": "String"`
		`}`

parms.additionalParameters	Object	Optional. Array of JSON objects. Each object contains key-value pairs that provide additional information for performing the translation.
		`"additionalParameters": {`
		`  "parameterName": "String",`
		`  "parameterValue": "String"`
		`}`

Name	Type	Description
parms.additionalParameters.parameterName	String	Optional. Key name. Valid values: textype: Type of text to translate. For Microsoft Azure Translator Service only.
parms.additionalParameters.parameterValue	String	Optional. Value of the associated key. Valid values: • plain: Standard text string • html: HTML text string Default: plain
parms.sourceLanguage	String	Optional. Language code of the source text. Default: Translation service detects the source language.
parms.targetLanguages	Array	Optional. List of language codes to use to translate the text. The method returns translated text for each language code.

Name	Type	Description
		Default: User preferred language.
parms.translator	String	Optional. Translation service to use to translate the text (not case-sensitive). Valid values: • Google • Microsoft • IBM • <custom> **Note:** To use custom translation services you must first configure the translation service in your instance. For details, see Integrate with a translation service provider. Default: Translation service configured in the Translator Configuration [sn_dt_translator_configuration] table.

Returns

Type	Description
status	Status of the response to the method call. Possible values: • Error • Partial • Success Data type: String

translations	Array of objects that describe the language translations.
	
	Data type: Array
	
	`translations": [`
	`  {`
	`    "isError": Boolean;`
	`    "detectedLanguage": {Ob`
	`ject},`
	`    "textTranslations": [Ar`
	`ray]`
	`  }`
	`]`

translations.isError	Flag that indicates whether the translation of the text resulted in an error. Valid values: • true: Error encountered. • false: Text translation was successful.

Type	Description
	Data type: Boolean

translations.detectedLanguage	Description of the detected language.
	
	Data type: Object
	
	`"detectedLanguage": {`
	`  "code": "String",`
	`  "name": "String"`
	`}`

translations.detectedLanguage.code	Language code of the detected language. Data type: String
translations.detectedLanguage.name	Language code of the detected language. Data type: String

textTranlations	Array of objects. Description of the language used to translate the text string.
	
	`"textTranslation": {`
	`  "targetLanguage": "String",`
	`  "translatedText": "String"`
	`}`

textTranslations.targetLanguage	Language code to which the source text was translated. Data type: String

Type	Description
textTranslations.translatedText	Translated text. Data type: String
translator	Translation service used to translate the text. Data type: String
Error messages	The following are error messages that the method may return and indications as to their root cause. • Text ("text" field) is missing or invalid. (40000): The text to translate is either missing or not a string. • Dynamic Translation plugin is not installed. (40001): The Dynamic Translation API was invoked without activating the com.glide.dynamic_translation plugin. For information on activating this plugin, see Dynamic translation overview. • Default translator is not configured for translation. (40002): No translation service is selected as the default translation service in the Translator Configurations. For information on creating/ modifying a translator configuration, see Create a translator configuration. • Translator ("translator" field) is invalid. (40003): The passed in

Type	Description
	translator parameter is not a string. • <translator> translator is not configured. (40004): The specified translation service is not configured in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • <translator> translator is inactive. (40005): The specified translation service is not set to **Active** in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Additional parameters are invalid. (40006): The additional parameters that were passed are not an object. • Target languages ("targetLanguages" field) are invalid. (40007): The targetLanguages parameter is passed in the call but is not valid for one of the following reasons: • Value is not an array • Array is empty • One or multiple of the entries is not a string • Source language ("sourceLanguage" field)

Type	Description
	is invalid. (40008): The sourceLanguage parameter is passed in the call but the value is not a String. • Maximum time limit has been exceeded. (40009): The operation took longer than the defined timeout value specified in the Translation Configuration. Default: 40 seconds • Request failed with multiple errors. (40010): Multiple errors occurred in the language detection call. For more information, refer to the response for each individual text string. • <translator> translator is not configured for translation. (40012): The specified translation service is not configured for text translation in the Translator Configuration. For information on creating/modifying a translator configuration, see Create a translator configuration. • Translator configuration version is invalid. Migrate to v3. (40014): The associated version of the Translator Configuration for the specified translation service does not support the specified text translation method. For more information, see Migrate to version v3 of a translator configuration. • Unknown error occurred. (40051): Default error thrown when the

Type	Description
	error doesn’t fall in to any other category. • Text ("text" field) has exceeded its maximum length. (40052): The text that was passed in to translate exceeds the maximum length supported by the corresponding translation service. • Source language is invalid. (40053): The passed in sourceLanguage parameter contains a language code that is not supported by the corresponding translation service. • Target language is invalid. (40054): One or more of the language codes passed in the targetLanguages parameter is not supported by the corresponding translation service. • Request is not authorized because credentials are missing or invalid (40055): The credentials configured for the translation service in Connections & Credentials are not valid. For information on connections and credentials, see Dynamic translation overview. • Text cannot be translated to target languages. (40056): The specified translation service is not able to translate the passed

Type	Description
	in text into the specified target languages.

Example

This example shows code in a portal client script that translates a list of text strings into French and Italian using the Microsoft translation service.

```
dynamicTranslation.getTranslations(["Translate first text
using portal from client", "Translate second text using po
rtal from client"], {
  "targetLanguages": ["fr", "it"],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```
{
  "translations":[
    {
      "isError":false,
      "textTranslations":[
        {
          "targetLanguage":"it",
          "translatedText":"Tradurre il primo testo utiliz
zando il portale dal client"
        },
        {
          "targetLanguage":"fr",
          "translatedText":"Traduire le premier texte à l’
aide du portail à partir du client"
        }
      ],
      "detectedLanguage": {"name":"en", "code":"en"}
    },
    {
      "isError":false,
      "textTranslations":[
        {
          "targetLanguage":"it",
          "translatedText":"Traduci il secondo testo utili
zzando il portale dal client"
        },
        {
          "targetLanguage":"fr",
          "translatedText":"Traduire le deuxième texte à l
’aide du portail à partir du client"
        }
      ],
      "detectedLanguage": {"name":"en", "code":"en"}
    }
  ],
  "translator":"Microsoft",
  "status":"Success"
}
```

Example

This example shows a portal client script that returns a Partial status when one of the two passed in text strings is invalid. To use this code example for a platform client script, change `dynamicTranslation.getTranslations` to `DynamicTranslation.getTranslations`.

```
dynamicTranslation.getTranslations(["Translate first text
using portal from client", ""], {
  "targetLanguages": ["fr", "it"],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": "Microsoft"
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```
{
  "translations":[
    {
      "isError":false,
      "textTranslations":[
        {
          "targetLanguage":"it",
          "translatedText":"Tradurre il primo testo utiliz
zando il portale dal client"
        },
        {
          "targetLanguage":"fr",
          "translatedText":"Traduire le premier texte à l’
aide du portail à partir du client"
        }
      ],
      "detectedLanguage":{"name":"en", "code":"en"}
    },
    {
      "isError":true,
      "code":"40000",
      "message":"Text is missing or invalid"
    }
  ],
 "translator":"Microsoft",
 "status":"Partial"
}
```

Example

This example shows a portal client script that throws an error when an invalid translation service is passed in. To use this code example for a platform client script, change `dynamicTranslation.getTranslations` to `DynamicTranslation.getTranslations`.

```
dynamicTranslation.getTranslations(["Translate first text
using portal from client", "Translate second text using po
rtal from client"], {
  "targetLanguages": ["fr", "it"],
  "additionalParameters": [{
    "parameterName": "texttype",
    "parameterValue": "plain"
  }],
  "translator": 123
 }).then(function(res){console.log(res);}, function(res){c
onsole.log(res);});
```

Response:

```
{"code":"40003","message":"Translator (\"translator\" fiel
d) is invalid","status":"Error"}
```

DynamicTranslation - isEnabled(String translator)

Determines whether the various methods in the DynamicTranslation API are enabled for a translation service.

If you pass in a specific translation service, the method checks the method activation for that translation service; otherwise the method checks the default translation service.

When calling this method from a portal client script, use the class name dynamicTranslation; such as dynamicTranslation.isEnabled(). When calling it from a platform client script, use the class name DynamicTranslation; such as DynamicTranslation.isEnabled().

Parameters

Name	Type	Description
translator	String	Optional. Translation service to use to verify whether the methods are active. Translation services are configured under the Translator Configuration menu. Possible values - not case-sensitive: • Google • Microsoft • IBM

Name	Type	Description
		• <custom> **Note:** To use custom translation services you must first configure the translation service in your instance. For details, see Integrate with a translation service provider. Default: Default translation service.

Returns

Type	Description
batchDetection	Flag that indicates whether the getDetectedLanguages() method is enabled. Possible values: • true: Enabled • false: Disabled Data type: Boolean
batchTranslation	Flag that indicates whether the getTranslations() method is enabled. Possible values: • true: Enabled

Type	Description
	• false: Disabled Data type: Boolean
detection	Flag that indicates whether the getDetectedLanguage() method is enabled. Possible values: • true: Enabled • false: Disabled Data type: Boolean
translation	Flag that indicates whether the getTranslation() method is enabled. Possible values: • true: Enabled • false: Disabled Data type: Boolean
Error messages	The following are error messages that the API may return and indications as to their root cause. • Translator ("translator" field) is invalid. (40003): The passed in translator parameter is not a string.

Example

This example shows a client script that checks whether the DynamicTranslation methods are enabled for the Microsoft translation service. To use this code example for a platform client script, change `DynamicTranslation.getTranslations` to `dynamicTranslation.getTranslations`.

```
DynamicTranslation.isEnabled('Microsoft').then(function(re
s){console.log(res);}, function(res){console.log(res);});
```

Output:

```
{"detection":true,"batchTranslation":true,"batchDetection"
:true,"translation":true}
```

Example

This example shows a client script that throws an error when an invalid translation service is passed in. To use this code example for a platform client script, change `DynamicTranslation.getTranslations` to `dynamicTranslation.getTranslations`.

```
DynamicTranslation.isEnabled(123).then(function(res){conso
le.log(res);}, function(res){console.log(res);});
```

Output:

```
{"code":"40003","message":"Translator (\"translator\" fiel
d) is invalid"}
```

g_service_catalog - Client

The g_service_catalog API enables you to access data in a multi-row variable set (MRVS) when a model is open.

This API is available in all environments, such as, Service Portal, Now Platform, Workspace, and Now® Mobile.

g_service_catalog - getValue(String variableName)

Returns the value of the specified field on the catalog item form when used in a client script on multi-row variable sets (MRVS).

Use this method when an MRVS modal is open for editing or creating and you want to modify data within the MRVS based on the value of a field on the parent catalog item form. For example, when you need to modify the contents of the cells within an MRVS based on a check box on the parent form. You can also use this method to access the data of other MRVS elements within the same parent form.

Note: This method can only be called from the parent object, such as g_service_catalog.parent.getValue().

Parameters

Name	Type	Description
variableName	String	Name of the variable in the catalog item form to return.

Returns

Type	Description
Array	Array of objects, with each object containing the name-value pairs of the requested variable. For example: `[{"city": "San Diego", "country": "USA"}, {"city": "New Delhi", "country": "India"}, {"city": "Melbourne", "country": "Australia"}]`

Example

In this example, a catalog item for blocking multiple IP addresses on a firewall has a variable address_type with two choices - **IPV4** and **IPV6**. The MRVS has two variables (ipv4_address and ipv6_address) for the respective address types. If the **Address type** field on the parent form is set to **IPV4**, the field **IPV6 address** is hidden on the MRVS.

```
function onLoad() {
  if (g_service_catalog.parent.getValue("address_type") =
= "ipv4") {
    g_form.setValue("ipv4_address", "XX.XX.XX.XX");
    g_form.setVisible("ipv6_address", "false");
  }
}
```

GlideAgentWorkspace (g_aw) - Client

The g_aw API enables a UI Action or client script to open a specified record in an Agent Workspace tab.

There is no constructor for the GlideAgentWorkspace class. Access GlideAgentWorkspace methods using the g_aw global object.

GlideAgentWorkspace - closeRecord()

Closes the currently open record, such as a form, in a subtab within Agent Workspace.

Parameters

Name	Type	Description
None		

Returns

Type	Description
None	

Example

The following example saves the content of the tab and then closes it.
`function onClick(g_form) {`

```
function onClick(g_form) {
  g_form.save().then(function(){
    g_aw.closeRecord();
  });
}
```

GlideAgentWorkspace - openRecord(String table, String sysId, Object params)

Opens a specified record, such as a form, in a subtab within Agent Workspace.

> **Note:** This method is only available in the Agent Workspace client scripting environment or in a UI action on the workspace client script field.

Parameters

Name	Type	Description
table	String	Name of the table that contains the record to open.
sysId	String	Sys ID of the record to open.

params	Object	Optional. Name/value pairs of the parameters to pass to the record.
		 `"params": {`
		`  "readOnlyForm" : Boolean;`
		`  "defaultTab": " String";`
		`  "hideDetails": Boolean`
		`}`

params.readOnlyForm	Boolean	Flag that indicates whether all fields on the opened record are read-only;

Name	Type	Description
		regardless of the UI policy and ACLs. • true: All fields are read only. • false: Fields adhere to the associated UI policy and ACLs. Default: false
params.defaultTab	String	Name of the initial tab to display in the workspace. You can only specify related items or related lists. If not specified, the details tab appears unless hideDetails is set to true. For more information on the method to use to obtain a related list name, see getRelatedListNames() .
params.hideDetails	Boolean	Flag that indicates whether to hide the details tab and the UI actions. • true: Only the form header, all other tabs, and the first available tab appear on the form.

Name	Type	Description
		• false: Details tab and UI actions appear on the form. Default: false

Returns

Type	Description
None	

Example

Open a sys_user record in a subtab.

```
g_aw.openRecord('sys_user', '62826bf03710200044e0bfc8bcbe5
df1');
```

Example

Open a record in a subtab where all fields are read-only.

```
g_aw.openRecord('sys_user', '62826bf03710200044e0bfc8bcbe5
df1', {readOnlyForm: true});
```

Example

Open a record in a subtab and go directly to the "Groups" related list.

```
g_aw.openRecord('sys_user', '62826bf03710200044e0bfc8bcbe5
df1', {defaultTab: "sys_user_grmember.user"});
```

Example

Open a record in a subtab but only show the form header and other tabs.

```
g_aw.openRecord('sys_user', '62826bf03710200044e0bfc8bcbe5
df1', {hideDetails: true});
```

GlideAjax - Client

The GlideAjax class enables a client script to call server-side code in a script include.

To use GlideAjax in a client script, follow these general steps.

1. Create a GlideAjax instance by calling the GlideAjax constructor. As the argument to the constructor, specify the name of the script include class that contains the method you want to call.
2. Call the addParam method with the sysparm_name parameter and the name of the script-include method you want to call.
3. (Optional) Call the addParam method one or more times to provide the script-include code with other parameters it needs.
4. Execute the server-side code by calling getXML().

    Note: getXML() is the preferred method for executing the code, because it is asynchronous and does not hold up the execution of other client code. Another method, getXMLWait(), is also available but is not recommended. Using getXMLWait() ensures the order of execution, but can cause the application to seem unresponsive, significantly degrading the user experience of any application that uses it. getXMLWait() is not available to scoped applications.

Example

```
var ga = new GlideAjax('HelloWorld'); // HelloWorld is th
e script include class
ga.addParam('sysparm_name','helloWorld'); // helloWorld i
s the script include method
ga.addParam('sysparm_user_name',"Bob"); // Set parameter s
ysparm_user_name to 'Bob'
ga.getXML(HelloWorldParse);  /* Call HelloWorld.helloWorld
() with the parameter sysparm_user_name set to 'Bob'
      and use the callback function HelloWorldParse() to r
eturn the result when ready */

// the callback function for returning the result from th
e server-side code
function HelloWorldParse(response) {
   var answer = response.responseXML.documentElement.getAt
tribute("answer");
    alert(answer);
}
```

GlideAjax - GlideAjax(String class_name)

Constructor for GlideAjax.

Parameters

Name	Type	Description
class_name	String	The name of the server-side class that contains the method you want to execute.

Example

In this example, a client script sets the user to Fred Luddy and then calls a script include to get their manager.

```
// client script - contains onLoad function and a callback
 function

function onLoad() {
   var ga = new GlideAjax('GetUserInfo'); // GetUserInfo i
s the script include name
   ga.addParam('sysparm_name','managerName'); // managerNa
me is the function in the script include that we’re callin
g
   ga.addParam('sysparm_user_name','fred.luddy'); // set u
ser to Fred Luddy

   /* Call GetUserInfo.managerName() with user set to Fred
 Luddy and use the callback function ManagerParse() to ret
urn the result when ready */
   ga.getXMLAnswer(ManagerParse);
}

// callback function for returning the result from the scr
ipt include
```

```
function ManagerParse(response) {
   alert(response);
}


// GetUserInfo script include

var GetUserInfo = Class.create();
GetUserInfo.prototype = Object.extendsObject(global.Abstra
ctAjaxProcessor, {

    managerName: function() {
        var userName = this.getParameter("sysparm_user_nam
e");
        var grUser = new GlideRecord('sys_user');
        grUser.get("user_name", userName);

        // Build the payload. You can return additional da
ta if needed.
        var result = {
           "manager": grUser.getDisplayValue('manager')
        };
        return JSON.stringify(result);
    },
    type: 'GetUserInfo'
});
```

GlideAjax - addParam(String parm_name, String parm_value)

Specifies a parameter name and value to be passed to the server-side function associated with this GlideAjax object.

You can execute addParam multiple times with different parameters and values.

Note: The first call to addParam should be with the parameter sysparm_name and the name of the server-side method you want to call. The server-side code does not execute until the client script calls getXML() or getXMLAnswer().

Parameters

Name	Type	Description
parm_name	String	The name of the parameter to pass. (The name must begin with the `sysparm_`.)
parm_value	String	The value to assign to parm_name.

Returns

Type	Description
void	

Example

In this example, a client script sets the user to Fred Luddy and then calls a script include to get their manager.

```
// client script - contains onLoad function and a callback
 function

function onLoad() {
   var ga = new GlideAjax('GetUserInfo'); // GetUserInfo i
s the script include name
   ga.addParam('sysparm_name','managerName'); // managerNa
me is the function in the script include that we're callin
g
   ga.addParam('sysparm_user_name','fred.luddy'); // set u
ser to Fred Luddy

   /* Call GetUserInfo.managerName() with user set to Fred
 Luddy and use the callback function ManagerParse() to ret
urn the result when ready */
   ga.getXMLAnswer(ManagerParse);
}

// callback function for returning the result from the scr
```

```
ipt include
function ManagerParse(response) {
   alert(response);
}


// GetUserInfo script include

var GetUserInfo = Class.create();
GetUserInfo.prototype = Object.extendsObject(global.Abstra
ctAjaxProcessor, {

    managerName: function() {
        var userName = this.getParameter("sysparm_user_nam
e");
        var grUser = new GlideRecord('sys_user');
        grUser.get("user_name", userName);

        // Build the payload. You can return additional da
ta if needed.
        var result = {
           "manager": grUser.getDisplayValue('manager')
        };
        return JSON.stringify(result);
    },
    type: 'GetUserInfo'
});
```

GlideAjax - getAnswer()

Retrieves the results from a server-side method called from the client via getXMLWait().

Parameters

Name	Type	Description
none		

Returns

Type	Description
void	The result returned by the server-side method previously called with getXMLWait().

GlideAjax - getXMLAnswer(Function callback, Object additionalParam, Object responseParam)

Calls the processor asynchronously and gets the answer element of the response in XML format.

Parameters

Name	Type	Description
callback	Function	Callback function. The function receives the answer element of the response in XML format as an argument.
additionalParam	Object	Optional. Name-value pair of additional parameters.
responseParam	Object	Optional. Second argument for the callback function.

Returns

Type	Description
void	

Example

Returns the number of attachments

```
function updateAttachmentCount(sysid) {
  var ga = new GlideAjax('AttachmentAjax');
  ga.addParam('sysparm_type', 'attachmentCount');
  ga.addParam('sysparm_value', sysid);
  ga.getXMLAnswer(numberOfAttachments, null, sysid); // ca
llback: numberOfAttachments with args (answer, sysid)
}

function numberOfAttachments(answer, sysid) {
  // we want to know there are 5 attachments, not 5.0 atta
chments
  var number = parseInt(answer);
  var buttons = $$('.attachmentNumber_' + sysid);
  if (buttons[0] == undefined)
    $('header_attachment_list_label').down().innerHTML = n
umber;
  else {
    for (var i = 0; i < buttons.length; i++) {
      buttons[i].innerHTML = number;
    }
  }
}
```

GlideAjax - getXML(Function callback)

Sends the server a request to execute the method and parameters associated with this GlideAjax object.

The server processes the request asynchronously and -- when ready -- returns the results via the function specified as the callback_function.

Parameters

Name	Type	Description
callback	Function	The name of the callback function to process the results returned by the server.

Returns

Type	Description
void	

Example

```
var comments = gel("dialog_comments").value;
var ga = new GlideAjax('validateComments'); //Call script
include to escape text
ga.addParam('sysparm_name', 'validateComments');
ga.addParam('sysparm_comments', comments);
ga.getXML(callback);

return false;

function callback(response) {
  var comments = response.responseXML.documentElement.getA
ttribute("answer");
  comments = trim(comments);
  if (comments == "") {
     //If comments are empty, alert the user and stop subm
ission
     alert("Please enter your comments before submitting."
);
  } else {
    //If there are comments, close the dialog window and s
ubmit them
    GlideDialogWindow.get().destroy(); //Close the dialog
window
    g_form.setValue("comments", comments); //Set the "Comm
ents" field with comments in the dialog
  }
```

GlideAjax - getXMLWait()

Sends the server a request to execute the method and parameters associated with this GlideAjax object.

The server processes the request synchronously and will not process further requests from the client until finished. To retrieve the results, the client must call getAnswer(). Using getXMLWait() ensures the order of execution, but can cause the application to seem unresponsive, significantly degrading the user experience of any application that uses it. We recommend using getXML() instead.

Note: getXMLWait() is not available to scoped applications.

Parameters

Name	Type	Description
none		

Returns

Type	Description
void	

Example

```
var ga = new GlideAjax('HelloWorld');
      ga.addParam('sysparm_name','helloWorld');
      ga.addParam('sysparm_user_name',"Bob");
      ga.getXMLWait();
      alert(ga.getAnswer());
```

Scoped equivalent

getXMLWait() is not available to scoped applications. Instead use the method.

GlideDialogWindow - Client

The GlideDialogWindow API provides methods for displaying a dialog in the current window and frame.

Use these methods in scripts anywhere that you can use a client-side JavaScript. These methods are most often called from a UI action with the **Client** check box selected.

Note: This API has been deprecated, use the GlideModalV3 API instead.

GlideDialogWindow - GlideDialogWindow(String id, Boolean readOnly, Number width, Number height)

Provides methods for displaying a dialog in the current window and frame.

Use these methods in scripts anywhere that you can use a client-side JavaScript. These methods are most often called from a UI Action with the **Client** check box selected.

Parameters

Name	Type	Description
id	String	Name of the UI page to load into the dialog window.
readOnly	Boolean	Optional. Flag that indicates whether the dialog window is read only (true) or read/write (false). Default: false
width	Number	Optional. Size (in pixels) to set the width of the dialog window.
height	Number	Optional. Size (in pixels) to set the height of the dialog window.

Returns

Type	Description
void	

Example

```
// Creates a dialog window
var gdw = new GlideDialogWindow('show_list');

// Creates a read-only dialog window
var gdw = new GlideDialogWindow('show_list', true);

// Creates a dialog window that is 400 pixels wide
var gdw = new GlideDialogWindow('show_list', false, 400);

// Creates a dialog window that is 400 pixels wide and 20
0 pixels tall
var gdw = new GlideDialogWindow('show_list', false, 400, 2
00);
```

GlideDialogWindow - adjustBodySize()

Adjusts the body height of a dialog window to be the window height minus the header height.

You typically call this method after calling GlideDialogWindow - setSize().

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
var gdw = new GlideDialogWindow('show_list');
      gdw.setTitle('Test');
      gdw.setSize(750,300);
      gdw.adjustBodySize();
      gdw.render();
```

GlideDialogWindow - destroy()

Closes the dialog window.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
//Destroy the current dialog window.
      GlideDialogWindow.get().destroy();
```

GlideDialogWindow - render()

Renders the dialog window.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
var gdw = new GlideDialogWindow('show_list');
      gdw.setTitle('Test');
      gdw.setSize(750,300);
      gdw.setPreference('table', 'u_test_list');
      gdw.setPreference('title', 'A New Title');
      gdw.render();
```

GlideDialogWindow - setPreference(String name, String value)

Sets a given window property to a specified value.

Any window property can be set using this method.

Parameters

Name	Type	Description
name	String	The window property to set.
value	String	The value for the window property.

Returns

Type	Description
void	

Example

```
var gdw = new GlideDialogWindow('show_list');
      gdw.setTitle('Test');
      gdw.setSize(750,300);
      gdw.setPreference('table', 'u_test_list');
      gdw.setPreference('title', 'A New Title');
```

GlideDialogWindow - setSize(Number width, Number height)

Sets the size of the dialog window.

If you do not pass width and height parameters, a default size is used.

Parameters

Name	Type	Description
width	Number	The width of the dialog window.
height	Number	The height of the dialog window.

Returns

Type	Description
void	

Example

```
var gdw = new GlideDialogWindow('show_list');
      gdw.setSize(750,300);
```

GlideDialogWindow - setTitle(String title)

Sets the title of the dialog window.

Parameters

Name	Type	Description
title	String	The title for the current window.

Returns

Type	Description
void	

Example

```
//var gdw = new GlideDialogWindow('show_list');
      gdw.setTitle('test');
```

GlideDocument - Client

The GlideDocument class provides the ability to search a DOM element, a document, or a JQuery element.

The GlideDocumentV3 API can be used in client-side scripts using ListV2 and ListV3 APIs. The GlideDocument APIs are accessed using the g_document global object.

GlideDocument - getElement(String selector, Element context)

Returns a node found in the specified DOM based context or created from the HTML context.

Parameters

Name	Type	Description
selector	String or Object	Selector expression
context	String or Object	Optional. DOM Element, document, or JQuery object to search.

Returns

Type	Description
node	Node that matches the selector.

Example

This example shows how to get the list view name value from a UI page.

```
//HTML entry
<input type="hidden" id="list_view" name="list_view" value
="${sysparm_list_view}" />

// Client script
var listView = g_document.getElement('#list_view').value;
```

GlideDocument - getElements(String selector, Element context)

Returns a node list found in the specified DOM based context or created if an HTML context is specified.

Parameters

Name	Type	Description
selector	String or Object	The selector expression
context	String or Object	(Optional) A DOM Element, document, or JQuery object to be searched.

Returns

Type	Description
node list	A list of nodes that matches the selector.

GlideFlow - Client

Use the GlideFlow JavaScript API for client-side interactions with actions, flows, and subflows.

Use the GlideFlow API anywhere in the platform that accepts client scripts. The action, flow, or subflow must be set as client callable, and have a valid ACL using the Manage Security feature in Flow designer.

Some of the methods within GlideFlow return `promise` objects. A `promise` represents the eventual result of an asynchronous operation. For more information on promises, see Promise - Javascript MDN or AngularJS documentation.

Using this API, you can:

- Start actions, flows, or subflows via a script.
- Get an existing execution.

- Get the status and any available outputs.
- Wait for the completion of an action, flow, or subflow.

There is no constructor for the GlideFlow class. Access GlideFlow methods using the GlideFlow global object.

GlideFlow - execution.awaitCompletion()

Returns a completion object for the execution.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Object	An object that contains completion details for a flow or action execution.

Example

In this example, an action is executed using startAction(), which returns an execution object. The code then uses awaitCompletion() on this execution object, which returns a completion object. The code uses this completion object to log the status and outputs within the execution.

```
(function() {
        var inputs = {};

        inputs['input1'] = 'string input'; // String

        GlideFlow.startAction('global.action_name', inputs
)
                .then(function(execution) {
                        return execution.awaitCompletion()
;
```

```
                }, errorResolver)
                .then(function(completion) {
                        var status = completion.status;
                        console.log(status);

                        // Available Outputs:
                        var outputs = completion.outputs;
                        console.log(outputs);
                }, errorResolver());

        function errorResolver(error) {
                // Handle errors in error resolver
                console.error(error);
        }
})();
```

GlideFlow - execution.getExecutionStatus()

Returns a string containing the execution status of the current execution.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	A string containing the execution status.

Example

In this example, the code obtains an execution object using the getExecution method. The getExecution method requires an ID, which is returned by the method used to start the execution. The code then uses getExecutionStatus() to determine whether the execution has been completed before continuing.

```
// Get an existing action, getStatus, and getOutputs if co
```

```
mplete
(function() {
   GlideFlow.getExecution('mamIN4Q35vmEFe744EwJV5GHrSz8fmJ
G')
      .then(function(execution) {
         execution.getExecutionStatus().then(
            function(status) {
               if (status === 'COMPLETE')
                  execution.getOutputs().then(
                     function(outputs) {
                        console.log(outputs);
                     },
                     errorResolver
                  );
            },
            errorResolver
         );
      }, errorResolver);

   function errorResolver(error) {
      // Handle errors in error resolver
      console.error(error);
   }
})();
```

GlideFlow - execution.getOutputs()

Returns an outputs object for the execution.

Use this method to access output generated by the execution of an action, flow, or subflow.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Object	An object containing outputs for an action, flow, or subflow.

Example

In this example, the code obtains an execution object using the getExecution method. After the execution is complete, the code uses getOutputs() to return an outputs object, which it then logs using the console.log method.

```
// Get an existing action, getStatus, and getOutputs if co
mplete
(function() {
   GlideFlow.getExecution('mamIN4Q35vmEFe744EwJV5GHrSz8fmJ
G')
      .then(function(execution) {
         execution.getExecutionStatus().then(
            function(status) {
               if (status === 'COMPLETE')
                  execution.getOutputs().then(
                     function(outputs) {
                        console.log(outputs);
                     },
                     errorResolver
                  );
            },
            errorResolver
         );
      }, errorResolver);

   function errorResolver(error) {
      // Handle errors in error resolver
      console.error(error);
   }
})();
```

GlideFlow - getExecution(String executionId)

Get an existing execution instance by ID.

Parameters

Name	Type	Description
executionId	String	The ID of the execution to be retrieved.

Returns

Type	Description
Object	A promise of an execution object.

Example

In this example, the code gets an execution, then waits for it to be completed before logging the executions completion status and outputs using console.log.

```
// Get an existing action and await completion
(function() {
        GlideFlow.getExecution('79cd437e0b202300a150a95e93
673ae3')
                .then(function(execution) {
                        return execution.awaitCompletion()
;
                }, errorResolver)
                .then(function(completion) {

                        var status = completion.status;
                        console.log(status);

                        // Available Outputs:
                        var outputs = completion.outputs;
                        console.log(outputs);
                }, errorResolver());
        function errorResolver(error) {
                // Handle errors in error resolver
                console.error(error);
        }
})();
```

GlideFlow - startAction(String scopedName.actionName, Map inputs)

Start an action.

Parameters

Name	Type	Description
scopedName	String	The scoped name of the flow to be executed.
inputs	Object	An object containing inputs defined for the action.

Returns

Type	Description
Object	An object containing details on the action execution.

Example

In this example, the code starts the global action_name action using arguments in the inputs input object variable. Upon completion, the example uses console.log or console.error to report on the success or failure of the flow.

```

// Start an action and await completion.
(function() {
        var inputs = {};

        inputs['input1'] = 'string input'; // String
        GlideFlow.startAction('global.action_name', inputs
)
                .then(function(execution) {
                        return execution.awaitCompletion()
;
                }, errorResolver)
                .then(function(completion) {
                        var status = completion.status;
                        console.log(status);

                        // Available Outputs:
                        var outputs = completion.outputs;
                        console.log(outputs);
                }, errorResolver());

        function errorResolver(error) {
                // Handle errors in error resolver
                console.error(error);
        }
})();
```

GlideFlow - startFlow(String scopedName.flowName, Map inputs)

Start a flow.

Parameters

Name	Type	Description
scopedName	String	The scoped name of the flow to be executed.
inputs	Object	An object containing inputs defined for the flow.

Returns

Type	Description
Object	An object containing details on the flow execution.

Example

This example flow is normally triggered when a record on the incident table is updated. Because you are activating the flow from Client script, you must provide this information. The code creates an inputs variable that contains the current record and the table for the record

```
// Start a Flow
(function() {
      var inputs = {};
      inputs['current'] = { // GlideRecord
        table : 'incident',
        sys_id : '79cd437e0b202300a150a95e93673ae3'
    };
        inputs['table_name'] = 'incident';
         GlideFlow.startFlow('global.flow_name', inputs)
                .then(
                        function(execution) {
                                console.log('Started flow_
name with execution id :' + execution.getExecutionId());
                        },
                        function(error) {
                                console.log('Unable to sta
rt flow: ' + error);
                        }
                );
})();
```

GlideFlow - startSubflow(String scopedName.subflowName, Map inputs)

Start a subflow.

Parameters

Name	Type	Description
scopedName	String	The scoped name of the flow to be executed.
inputs	Object	An object containing inputs used for the subflow.

Returns

Type	Description
Object	An object containing details on the subflow execution.

Example

In this example, the code starts the global subflow_name subflow using arguments in the inputs array variable. Upon completion, the example uses console.log or console.error to report on the success or failure of the flow.

```
// Start an action and await completion.
(function() {
        var inputs = {};

        inputs['input1'] = 'string input'; // String

        GlideFlow.startSubflow('global.subflow_name', inpu
ts)
                .then(function(execution) {
                        return execution.awaitCompletion()
;
                }, errorResolver)
                .then(function(completion) {
                        var status = completion.status;
                        console.log(status);
                        // Available Outputs:
                        var outputs = completion.outputs;
                        console.log(outputs);
                }, errorResolver());

        function errorResolver(error) {
                // Handle errors in error resolver
                console.error(error);
        }
})();
```

GlideForm - Client

The GlideForm API provides methods to customize forms.

GlideForm.js is the JavaScript class containing the methods. The global object g_form is used to access GlideForm methods. GlideForm methods are only used on the client. These methods are used to make custom changes to the form view of records. All validation of examples was done using client scripts.

Some of these methods can also be used in other client scripts (such as Catalog Client Scripts or Wizard Client Scripts), but must first be tested to determine whether they will work as expected.

Note: The methods getControl(), getHelpTextControl(), getElement(), and getFormElement() are deprecated for mobile devices. For information on using GlideForm for mobile, see Mobile Client GlideForm (g_form) Scripting and Migration .

There is no constructor for the GlideForm class. Access GlideForm methods using the g_form global object.

GlideForm - addDecoration(String fieldName, String icon, String title)

Adds an icon on a field’s label.

Adding the same item twice is prevented; however, you can add the same icon with a different title.

Note: This method is not supported by Service Catalog.

Parameters

Name	Type	Description
fieldName	String	The field name.
icon	String	The font icon to show next to the field. Supported icons - icon-user, icon-user-group, icon-lightbulb, icon-home, icon-mobile, icon-comment, icon-mail, icon-locked, icon-database, icon-book, icon-drawer, icon-folder, icon-catalog, icon-tab, icon-cards, icon-tree-right, icon-tree, icon-book-open, icon-paperclip, icon-edit, icon-trash, icon-image, icon-search, icon-power, icon-cog, icon-star, icon-star-empty, icon-new-ticket, icon-dashboard, icon-cart-full, icon-view, icon-label, icon-filter, icon-calendar, icon-script, icon-add, icon-delete, icon-help, icon-info, icon-check-circle, icon-alert, icon-sort-ascending, icon-console, icon-list, icon-form, and icon-livefeed.

Name	Type	Description
title	String	The text title for the icon.

Returns

Type	Description
void	

Example

```
g_form.addDecoration('caller_id', 'icon-star', 'preferred
member');
```

GlideForm - addDecoration(String fieldName, String icon, String title, String color)

Adds an icon on a field’s label.

Adding the same item twice is prevented; however, you can add the same icon with a different title.

Note: This method is not supported by Service Catalog.

Parameters

Name	Type	Description
fieldName	String	The field name.
icon	String	The font icon to show next to the field. Supported icons - icon-user, icon-user-group, icon-lightbulb, icon-home, icon-mobile, icon-comment, icon-mail, icon-locked, icon-database, icon-book, icon-drawer, icon-

Name	Type	Description
		folder, icon-catalog, icon-tab, icon-cards, icon-tree-right, icon-tree, icon-book-open, icon-paperclip, icon-edit, icon-trash, icon-image, icon-search, icon-power, icon-cog, icon-star, icon-star-empty, icon-new-ticket, icon-dashboard, icon-cart-full, icon-view, icon-label, icon-filter, icon-calendar, icon-script, icon-add, icon-delete, icon-help, icon-info, icon-check-circle, icon-alert, icon-sort-ascending, icon-console, icon-list, icon-form, and icon-livefeed.
title	String	The text title for the icon.
color	String	A CSS color.

Returns

Type	Description
void	

Example

```
g_form.addDecoration('caller_id', 'icon-star', 'Mark as Fa
vorite', 'color-green');
```

GlideForm - addErrorMessage(String message)

Displays the specified error message at the top of the form.

This message appears for approximately four seconds and then disappears. This timeout is not configurable at this time.

Parameters

Name	Type	Description
message	String	Message to display.

Returns

Type	Description
void	

Example

```
g_form.addErrorMessage('This is an error');
```

GlideForm - addFormMessage(String message, String type)

Displays a floating form message at the top of the form detail section. The message does not cover UI actions.

See also:

- clearAllFormMessages()
- clearFormMessages()

Parameters

Name	Type	Description
message	String	Message to display.
type	String	The type of message. Valid values:

Name	Type	Description
		• error • info • warning

Returns

Type	Description
None	

Example

The following example shows how to add form messages of each type.

```
g_form.addFormMessage('info message','info');
g_form.addFormMessage('warning message','warning');
g_form.addFormMessage('error message','error');
g_form.addFormMessage('info2 message','info');
g_form.addFormMessage('warning2 message','warning');
g_form.addFormMessage('error2 message','error');
```

GlideForm - addInfoMessage(String message)

Adds the specified informational message to the top of the form.

This message appears for approximately four seconds and then disappears. This timeout is not configurable at this time.

Parameters

Name	Type	Description
message	String	Message to display.

Returns

Type	Description
void	

Example

```
g_form.addInfoMessage('The top five fields in this form ar
e mandatory');
```

GlideForm - addOption(String fieldName, String choiceValue, String choiceLabel)

Adds a choice to the end of a choice list field.

Parameters

Name	Type	Description
fieldName	String	The name of the field.
choiceValue	String	The value to be stored in the database.
choiceLabel	String	The value displayed.

Returns

Type	Description
void	

Example

```
g_form.addOption('priority', '6', '6 - Really Low');
```

GlideForm - addOption(String fieldName, String choiceValue, String choiceLabel, Number choiceIndex)

Adds a choice to the list field at the position specified.

Note: Duplicate list labels are not supported in Service Portal. For example, items with label text matching another label are ignored and not added to the list.

Parameters

Name	Type	Description
fieldName	String	The field name.
choiceValue	String	The value stored in the database.
choiceLabel	String	The value displayed.
choiceIndex	Number	Order of the choice in the list. The index is into a zero based array.

Returns

Type	Description
void	

Example

```
g_form.addOption('priority', '2.5', '2.5 - Moderately High
', 3);
```

GlideForm - clearMessages()

Removes all informational and error messages from the top of the form.

Removes informational and error messages added with g_form.addInfoMessage() and g_form.addErrorMessage().

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
g_form.clearMessages();
```

GlideForm - clearAllFormMessages()

Removes all form messages of any type.

See also:

- addFormMessage()
- clearFormMessages()

Parameters

Name	Type	Description
None		

Returns

Type	Description
None	

Example

The following example shows how to clear all messages from the form.

```
g_form.clearAllFormMessages();
```

GlideForm - clearFormMessages(String type)

Removes all form messages of a given type.

See also:

- addFormMessage()
- clearAllFormMessages()

Parameters

Name	Type	Description
type	String	The type of message. Valid values: • error • info • warning

Returns

Type	Description
None	

Example

The following example shows how to clear all error messages from the form.

```
g_form.clearFormMessages('error');
```

GlideForm - clearOptions(String fieldName)

Removes all options from the choice list.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
void	

GlideForm - clearValue(String fieldName)

Removes any value(s) from the field.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
void	

GlideForm - disableAttachments()

Prevents file attachments from being added.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - enableAttachments()

Allows file attachments to be added. Shows the paper clip icon.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - flash(String fieldName, String color, Number count)

Used to draw attention to a particular field. Flashes the specified color for a specified duration of time in the specified field.

This method is not supported by Service Catalog.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
fieldName	String	Specifies the field to highlight in the following

Name	Type	Description
		format: "`<table-name>.<field-name>`".
color	String	RGB color or acceptable CSS color.
count	Number	Specifies how long the label will flash. Options include: • 2: Flashes for 1 second • 0: Flashes for 2 seconds • -2: Flashes for 3 seconds • -4: Flashes for 4 seconds

Returns

Type	Description
void	

Example

```
g_form.flash("incident.number", "#FFFACD", 0);
```

GlideForm - getActionName()

Returns the most recent action name, or, for a client script, the sys_id of the UI action clicked.

Note: Not available in Wizard client scripts.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The current action name.

Example

```
function onSubmit() {
   var action = g_form.getActionName();
   alert('You pressed ' + action);
}
```

GlideForm - getBooleanValue(String fieldName)

Returns a Boolean value for the specified field.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
Boolean	Returns false if the field value is false or undefined; otherwise returns true.

GlideForm - getControl(String fieldName)

Returns the HTML element for the specified field.

Compound fields may contain several HTML elements. This method is generally not necessary as there are built-in methods that use the fields on a form.

If the field is a reference field and the control is a choice list, getControl() may not return a control as expected. In this case, use `sys_select.<table name>.<field name>`.

This method is not available in mobile scripts or Service Portal scripts.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
HTMLElement	The field's HTML element.

GlideForm - getDecimalValue(String fieldName)

Returns the decimal value of the specified field.

Parameters

Name	Type	Description
fieldName	String	The name of the field.

Returns

Type	Description
String	The decimal value of the specified field.

Example

```
function onChange(control, oldValue, newValue, isLoading)
{
   alert(g_form.getDecimalValue('percent_complete'));
}
```

GlideForm - getDisplayBox(String fieldName)

Gets the display value from a form in the core UI.

> **Note:** To get a display value from a form in Service Portal, use the getDisplayValue() method.

See also:

- getValue()
- Get the display value of a reference variable

Parameters

Name	Type	Description
fieldName	String	Name of the field from which you want to retrieve a value in the form.

Returns

Type	Description
None	

Example

```
var caller = g_form.getDisplayBox('caller_id').value;

var assignee = g_form.getDisplayBox('assigned_to').value;

if (caller == assignee)
{
    alert('in');
}
```

GlideForm - getDisplayValue(String fieldName)

Gets the display value from a form in Service Portal.

> **Note:** To get a display value from a form in the core UI, use the getDisplayBox() method.

See also:

- getValue()
- Get the display value of a reference variable

> **Note:** In the core UI, calling this method as `g_form.getDisplayValue()` without an argument returns the record display value rather than the display value of an individual field.

Parameters

Name	Type	Description
fieldName	String	Name of the field from which you want to retrieve a value in the form.

Returns

Type	Description
None	

Example

The following example shows how to get the display value of a reference variable in the core UI or Service Portal. The use case for this example is on the community site.

```
function onChange(control, oldValue, newValue, isLoading)
{
     if (isLoading || newValue == '') {
          return;
     }
     if(window == null){
          var valuePortal = g_form.getDisplayValue('reques
ter');
          alert('Portal->' + valuePortal);
     }
     else{
          var valueNative = g_form.getDisplayBox('requeste
r').value;
          alert('CoreUI->' + valueCoreUI);
     }
     //Type appropriate comment here, and begin script bel
ow
}
```

GlideForm - getElement(String id)

Returns the HTML element specified by the parameter.

Compound fields may contain several HTML elements. This method is generally not necessary as there are built-in methods that use the fields on a form.

This method is not available in mobile scripts or Service Portal scripts.

Parameters

Name	Type	Description
id	String	The field id.

Returns

Type	Description
HTMLElement	The field's HTML element.

GlideForm - getFormElement()

Returns the HTML element for the form.

This method is not available in mobile scripts or Service Portal scripts.

Parameters

Name	Type	Description
None		

Returns

Type	Description
HTMLFormElement	The HTML element for the form.

GlideForm - getHelpTextControl(String fieldName)

Returns the HTML element of the help text for the specified field.

This method is applicable to service catalog variables only.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
HTMLElement	Help text field's HTML element.

GlideForm - getIntValue(String fieldName)

Returns the integer value of the field.

Parameters

Name	Type	Description
fieldName	String	The field name.

Returns

Type	Description
Number	Integer value of the field.

GlideForm - getLabelOf(String fieldName)

Returns the plain text value of the field label.

Parameters

Name	Type	Description
fieldName	String	The field name

Returns

Type	Description
String	The label text.

Example

```
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer v
isible)');
}
```

GlideForm - getOption(String fieldName, String choiceValue)

Returns the option element for a selected box named fieldName where choiceValue matches the option value.

Note: This method does not work on read-only fields.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
choiceValue	String	Value of the option.

Returns

Type	Description
HTMLElement	The HTMLElement for the option. Returns null if the field or option is not found.

Example

The following example shows how to get the label for a choice list value.

```
// Get the label for a choice list value
// fieldName is 'category'

function onChange(control, oldValue, newValue, isLoading)
{
var choiceValue = g_form.getValue('category');
var choiceLabel = g_form.getOption('category', choiceValue
).text;
}
```

GlideForm - getReference(String fieldName, Function callBack)

Returns the GlideRecord for a specified field.

If a callback function is present, this routine runs asynchronously. The browser (and script) processing continues normally until the server returns the reference value, at which time, the callback function is invoked. If a callback function is not present, this routine runs synchronously and processing halts (causing the browser to appear to hang) while waiting on a server response.

Important: It is strongly recommended that you use a callback function.

Callback function support for ServiceCatalogForm.getReference is available.

Note: Using this method requires a call to the server which requires additional time and may introduce latency to your page. Use this method with caution. For additional information, see Client script design and processing.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
callBack	Function	Name of the call back function.

Returns

Type	Description
GlideRecord	GlideRecord object for the specified field. If the specified reference cannot be found, it returns an initialized GlideRecord object where currentRow = -1 and rows.length = 0.

Example

```
function onChange(control, oldValue, newValue, isLoading)
{
    g_form.getReference('caller_id', doAlert); // doAlert
is our callback function
}
```

```
function doAlert(caller) { // reference is passed into cal
lback as first arguments
   if (caller.getValue('vip') == 'true') {
      alert('Caller is a VIP!');
   }
}
```

GlideForm - getRelatedListNames()

Returns an array of related list names from the current form.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Array	String array of related list names from the current form. The related list names are listed in the order in which they appear on the form.

Example

```
var listNames = g_form.getRelatedListNames();

for (var i = 0; i < listNames.length; i++) {
  this.showRelatedList(listNames[i]);
 }
```

GlideForm - getSectionNames()

Returns all section names, whether visible or not.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Array of strings	The section names.

GlideForm - getSections()

Returns an array of the form's sections.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Array of HTML elements	The form's sections.

Example

```
function onChange(control, oldValue, newValue, isLoading)
{
   //this example was run on a form divided into sections
(Change form)
   // and hid a section when the "state" field was changed
   var sections = g_form.getSections();
   if (newValue == '2') {
      g_form.setSectionDisplay(sections[1], false);
   } else {
        g_form.setSectionDisplay(sections[1], true);
    }
}
```

GlideForm - getTableName()

Returns the name of the table to which this record belongs.

On the server side, the table for the current record can be retrieved with current.sys_class_name or current.getTableName().

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Name of the table.

Example

```
function onLoad() {
    if (g_form.isNewRecord()) {
        var tableName = g_form.getTableName(); //Get the t
able name
    }
}
```

GlideForm - getUniqueValue()

Returns the sys_id of the record displayed in the form.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The record's sys_id.

Example

```
function onLoad() {
   var incSysid = g_form.getUniqueValue();
   alert(incSysid);
}
```

GlideForm - getValue(String fieldName)

Returns the value of the specified form field.

This method also supports getting values from a multi-row variable set (MRVS). To obtain data from fields within an MRVS, you must first use `JSON.parse(getValue('<mrvs_field_name>') || '[]')` to obtain the MRVS array, and then use indexing to access the fields within the row objects. For more details, see the code example below.

Parameters

Name	Type	Description
fieldName	String	Name of the field whose value to return.

Returns

Type	Description
String	Value of the specified field.

Example

The following example shows how to get the short description from the current form.

```
function onChange(control, oldValue, newValue, isLoading)
{
   alert(g_form.getValue('short_description'));
}
```

Example

The following example shows how to get values from an MVRS. In this example, salaries are being managed through the Service Catalog. The client script searches all rows within the MRVS for the value entered in the **Job title** and then updates the matching entries within the MVRS with what is entered in the **Salary** field. The MRVS is named "variable_set_1" and contains the following fields within each row object: Employee name [employee_name], Job title [employee_job_title], and Salary [employee_salary]. In addition, the Catalog Item contains: Job title [job_title] and Salary [salary].

```
function onChange(control, oldValue, newValue, isLoading)
{
if (isLoading || newValue == '') {
return;
}

// Get the MRVS
var multiRowVariableSet = JSON.parse(g_form.getValue('vari
able_set_1'));

for (var i = 0; i < multiRowVariableSet.length; i++) {
// Check if the entered job title matches the title in th
e current MRVS row
  if (multiRowVariableSet[i].employee_job_title == g_form.
getValue("job_title")){
    // Update the value of a matching field with the new s
alary
    multiRowVariableSet[i].employee_salary = newValue;
  }
}

// Update the MRVS
g_form.setValue('variable_set_1', JSON.stringify(multiRowV
ariableSet));
}
```

GlideForm - hideAllFieldMsgs()

Hides all field messages.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - hideAllFieldMsgs(String type)

Hides all field messages of the specified type.

Parameters

Name	Type	Description
type	String	The type of message. Valid values: • error • info

Returns

Type	Description
void	

GlideForm - hideErrorBox(String fieldName)

Hides the error message placed by showErrorBox().

Whenever possible, use hideFieldMsg() rather than this method whenever possible.

Parameters

Name	Type	Description
fieldName	String	The name of the field or control.

Returns

Type	Description
void	

GlideForm - hideFieldMsg(String fieldName)

Hides the last message placed by showFieldMsg().

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
void	

GlideForm - hideFieldMsg(String fieldName, Boolean clearAll)

Hides the messages placed by showFieldMsg().

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Name	Type	Description
clearAll	Boolean	When true, all messages for the field are cleared. When false, only the last message is removed.

Returns

Type	Description
void	

Example

```
g_form.hideFieldMsg('impact', true);
```

GlideForm - hideRelatedList(String listTableName)

Hides the specified related list on the form.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
listTableName	String	Name of the related list. Use the sys_id to hide a list through a relationship.

Returns

Type	Description
void	

GlideForm - hideRelatedLists()

Hides all related lists on the form.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - isLiveUpdating()

Returns true while a live update is being done on the record the form is showing.

This can be used in an onChange() client script to determine if a change to the record is because of a live update from another session. The client script can then decide what action to take, or not to take. This applies to systems using Core UI with live forms enabled.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	Returns true if a live update is happening on the record displayed by the form.

GlideForm - isMandatory(String fieldName)

Returns true if the field is mandatory.

Parameters

Name	Type	Description
fieldName	String	Name of the field.

Returns

Type	Description
Boolean	True if the field is required, false otherwise.

GlideForm - isNewRecord()

Returns true if the record has never been saved.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	Returns true if the record has not been saved; otherwise false.

Example

```
function onLoad() {
   if(g_form.isNewRecord()){
      alert('New Record!');
   }
}
```

GlideForm - isSectionVisible(String sectionName)

Returns true if the section is visible.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	Returns true when the section is visible; otherwise, false is returned.

GlideForm - onUserChangeValue(Function fn)

Registers a custom event listener that detects when any field in the current form is modified by a user.

When a form field is modified, the event listener calls the function that is passed in when the listener is initially registered. This listener is only triggered when a user makes a change to a field on the form. Changes from client scripts, UI policies, or any other non-user interactions, do not trigger the listener.

Note: This method does not work for journal fields.

Parameters

Name	Type	Description
fn	Function	Function to call when a user changes the value of a field within the current form. This is actually the function code, not just the function name.

Name	Type	Description
		This function must accept the following three arguments: • field name • original field value • updated field value

Returns

Type	Description
Function	Function to call to unregister the onUserChangeValue event listener.

Example

Example

```
var handler = function(fieldname, originalValue, newValue
) {
  console.log('The field ('+ fieldname + ') has a new valu
e of: ' + newValue); // function code
}

var unregister = g_form.onUserChangeValue(handler);

// To unregister the event listener
unregister();
```

GlideForm - refreshSlushbucket(String fieldName)

You can update a list collector variable.

Parameters

Name	Type	Description
fieldName	String	Name of the slush bucket.

Returns

Type	Description
void	

Example

```
g_form.refreshSlushbucket('bucket');
```

GlideForm - removeOption(String fieldName, String choiceValue)

Removes the specified option from the choice list.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
choiceValue	String	The value stored in the database. This is not the label.

Returns

Type	Description
void	

Example

```
g_form.removeOption('priority', '1');
```

GlideForm - removeDecoration(String fieldname, String icon, String title)

Removes the icon from the specified field that matches the icon and title.

Note: This method is not supported by Service Catalog.

Parameters

Name	Type	Description
fieldName	String	Field name.
icon	String	Name of the icon to remove.
title	String	The icon's text title (name).

Returns

Type	Description
void	

Example

```
function onChange(control, oldValue, newValue, isLoading)
{
        // if the caller_id field is not present, then we
can't add an icon anywhere
        if (!g_form.hasField('caller_id'))
                return;

        if (!newValue)
                return;

        g_form.getReference('caller_id', function(ref) {
                g_form.removeDecoration('caller_id', 'icon
-star', 'VIP');
```

```
                if (ref.getValue('vip') == 'true')
                        g_form.addDecoration('caller_id',
'icon-star', 'VIP');
        });
}
```

GlideForm - removeDecoration(String fieldname, String icon, String title, String color)

Removes the icon from the specified field that matches the icon, title, and color.

Note: This method is not supported by Service Catalog.

Parameters

Name	Type	Description
fieldName	String	Field name.
icon	String	Name of the icon to remove.
title	String	The icon's text title (name).
color	String	A CSS color

Returns

Type	Description
void	

Example

```
g_form.removeDecoration('caller_id', 'icon-star', 'VIP', '
blue');
```

GlideForm - save()

Saves the record without navigating away (update and stay).

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - setMandatory(String fieldName, Boolean mandatory)

Makes the specified field mandatory.

Whenever possible, use a UI policy rather than this method.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
mandatory	Boolean	When true makes the field mandatory. When false makes the field optional.

Returns

Type	Description
void	

GlideForm - setSectionDisplay(String sectionName, Boolean display)

Shows or hides a section.

Parameters

Name	Type	Description
sectionName	String	The section name is lower case with an underscore replacing the first space in the name, and with the remaining spaces being removed, for example "Section Four is Here" becomes "section_fourishere". Other non-alphanumeric characters, such as ampersand (&), are removed. Section names can be found by using the getSectionNames() method.
display	Boolean	When true shows the section. When false hides the section.

Returns

Type	Description
Boolean	Returns true when successful.

GlideForm - setValue(String fieldName, String value, String displayValue)

Sets the value of a specified form field to the value of a specified display value in a reference record.

To improve performance by preventing a round trip when setting the value for a reference field, use this method, not setValue(fieldName, value). When setting multiple reference values for a list collector field, pass arrays in the value and displayValue parameters.

Note: The method setValue() can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before performing `setValue(shortDesc, newValue.toUpperCase());`, validate that the short description is not already uppercase. This will prevent the client script from applying the toUpperCase() more than once.

Parameters

Name	Type	Description
fieldName	String	Name of the form field to update.
value	String or Array	Sys_id of the reference record to use to update the field. If the specified field is a GlideList, this parameter can contain an array of sys_ids. In this case, the method performs a lookup of all records specified in the array and those values are used to update the contents of the specified field (related list).

Name	Type	Description
		Note: When defining a value in a choice list, be sure to use number value rather than the label.
displayValue	String or Array	Field within the specified reference record to use to update the specified field. For example, in the User [sys_user] table it might be userName. If the specified field is a GlideList, this parameter can contain an array of display value names. For additional information on display values, see Display value.

Returns

Type	Description
void	

Example

This example shows passing the sys_id of the reference record that contains the userName field to use to update the **assigned_to** form field.

```
g_form.setValue('assigned_to', userSysID, userName);
```

Example

This example shows passing an array of reference record sys_ids and an array of corresponding display value names to use to update the form fields in the GlideList **glide-list_field_name**.

```
g_form.setValue('glide-list_field_name', sysIDArray, displ
ayNameArray);
```

GlideForm - showErrorBox(String name, String message, Boolean scrollForm)

Displays an error message under the specified form field (either a control object or the name of the field). If the control or field is currently off the screen and the scrollForm parameter is true, the form scrolls to the control or field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field). The showFieldMsg() method is a similar method that requires a type parameter.

Parameters

Name	Type	Description
name	String	Name of the field or control.
message	String	Message to display.
scrollForm	Boolean	When true scrolls the form to the field. When false the form does not scroll to the field.

Returns

Type	Description
void	

GlideForm - showFieldMsg(String field, String message, String type)

Displays either an informational or error message under the specified form field (either a control object or the name of the field). If the control or field is off the screen, the form is scrolled to the field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field).

The showErrorBox() method is a shorthand method that does not require the type parameter.

Note: This method does not work with the journal_field type field in Core UI.

Parameters

Name	Type	Description
field	String	Name of the field or control.
message	String	Message to display.
type	String	"error","info", or "warning".

Returns

Type	Description
void	

Example

```
g_form.showFieldMsg('impact','Low impact response time ca
n be one week','info');
```

GlideForm - showFieldMsg(String field, String message, String type, Boolean scrollForm)

Displays either an informational or error message under the specified form field (either a control object or the name of the field). If the control or field is currently off the screen and scrollForm is true, the form is scrolled to the field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field).

The showErrorBox() method is a shorthand method that does not require the type parameter.

Note: This method does not work with the journal_field type field in Core UI.

Parameters

Name	Type	Description
field	String	Name of the field or control.
message	String	Message to display.
type	String	"error","info", or "warning".
scrollForm	Boolean	When true, the form scrolls to the field if it is off screen. When false, the form does not scroll.

Returns

Type	Description
void	

Example

```
g_form.showFieldMsg('impact','Low impact not allowed with
High priority','error',false);
```

GlideForm - setDisabled(String fieldName, Boolean disable)

Makes the specified field available or unavailable.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
disable	Boolean	When true disables the field. When false enables the field.

Returns

Type	Description
void	

GlideForm - setDisplay(String fieldName, Boolean display)

Displays or hides a field.

This method cannot hide a mandatory field with no value. If the field is hidden, the space is used to display other items. Whenever possible, use a UI policy instead of this method.

Parameters

Name	Type	Description
fieldname	String	Name of the field.
display	Boolean	When true displays the field, when false hides the field.

Returns

Type	Description
void	

Example

```
function onChange(control, oldValue, newValue, isLoading,
isTemplate) {
   //If the page isn't loading
   if (!isLoading) {
      //If the new value isn't blank
      if (newValue != '') {
         g_form.setDisplay('priority', false);
      }
      else
         g_form.setDisplay('priority', true);
      }
   }
```

GlideForm - setLabelOf(String fieldName, String label)

Sets the plain text value of the field label.

Note: This method is not supported by Service Catalog.

Parameters

Name	Type	Description
fieldName	String	The field name.

Name	Type	Description
label	String	The field text label.

Returns

Type	Description
void	

Example

```
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer v
isible)');
}
```

GlideForm - setReadOnly(String fieldName, Boolean readOnly)

Makes the specified field read-only or editable.

Whenever possible, use a UI policy instead of this method.

To make a mandatory field read-only, you must first remove the mandatory requirement for that field by using the setMandatory() method.

Once you set a field to read-only, you cannot use the setValue() method to update the value of that field. If you need to set the value in this way, you must set the readOnly value to false.

Parameters

Name	Type	Description
fieldName	String	Name of the field.
readOnly	Boolean	Flag that determines whether the associate field is editable or read-only. Possible values:

Name	Type	Description
		• true: Set field to read-only • false: Set field to be editable

Returns

Type	Description
None	

GlideForm - setValue(String fieldName, String value)

Sets the value of a specified form field to the passed in value.

This method also supports setting values in a multi-row variable set (MRVS). You must first use `JSON.parse(getValue('<mrvs_field_name>'))` to obtain the MRVS array and then use indexing to update the fields within the row objects. Once all values are updated in the MRVS, use the setValue() method to save the updated MRVS array. For more details, see the code example below.

Note: The method setValue() can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before performing `setValue(shortDesc, newValue.toUpperCase());`, validate that the short description is not already uppercase. This will prevent the client script from applying the toUpperCase() more than once.

Note: When defining a value in a choice list, be sure to use the number value rather than the label.

Parameters

Name	Type	Description
fieldName	String	Name of the form field to update.
value	String	String value to set in the specified field.

Returns

Type	Description
void	

Example

The following example shows how to set the short description in the current form.

```
g_form.setValue('short_description', 'replace this with ap
propriate text');
```

Example

The following example shows how to set values in an MVRS. In this example, salaries are being managed through the Service Catalog. The client script searches all rows within the MRVS for the value entered in the **Job title** and then updates the matching entries within the MVRS with what is entered in the **Salary** field. The MRVS is named "variable_set_1" and contains the following fields within each row object: Employee name [employee_name], Job title [employee_job_title], and Salary [employee_salary]. In addition, the Catalog Item contains: Job title [job_title] and Salary [salary].

```
function onChange(control, oldValue, newValue, isLoading)
{
if (isLoading || newValue == '') {
return;
}
// Get the MRVS
var multiRowVariableSet = JSON.parse(g_form.getValue('vari
able_set_1'));

for (var i = 0; i < multiRowVariableSet.length; i++) {
// Check if the entered job title matches the title in th
e current MRVS row
  if (multiRowVariableSet[i].employee_job_title == g_form.
getValue("job_title")){
    // Update the value of a matching field with the new s
alary
    multiRowVariableSet[i].employee_salary = newValue;
  }
}

// Update the MRVS
g_form.setValue('variable_set_1', JSON.stringify(multiRowV
ariableSet));
}
```

GlideForm - setVariablesReadOnly(Boolean isReadOnly)

Makes a Service Catalog variable editor read only.

Note: This method is only applicable to Service Catalog variable editors in the core UI. This method is not supported in the Service Catalog form.

The method must be placed in the client script of the table in which the variable editor is added, such as Requested Item [sc_req_item], Incident [incident], and so on. To set variables to read only in other tables, use the setReadOnly() method.

See also: Service Catalog variable editors

Parameters

Name	Type	Description
isReadOnly	Boolean	Flag that determines whether the variable editor is read only.

Name	Type	Description
		Possible values: • true: Sets the variable editor as read-only. • false: Sets the variable editor as editable. Default: false

Returns

Type	Description
None	

Example

Adding the following line to a client script sets the variable editor to read only.

```
g_form.setVariablesReadOnly(true);
```

GlideForm - setVisible(String fieldName, Boolean display)

Displays or hides the field.

On desktop UI, the space is left blank when hidden. On Mobile or Service Portal UI, the space is filled in my other fields when hidden. This method cannot hide mandatory fields with no value.

Use UI Policy rather than this method whenever possible.

Parameters

Name	Type	Description
fieldName	String	The field name.

Name	Type	Description
display	Boolean	When true displays the field. When false hides the field.

Returns

Type	Description
void	

Example

```
function onChange(control, oldValue, newValue, isLoading,
isTemplate) {
   //If the page isn't loading
   if (!isLoading) {
      //If the new value isn't blank
      if(newValue != '') {
         g_form.setVisible('priority', false);
      }
      else
         g_form.setVisible('priority', true);
      }
   }
```

GlideForm - showErrorBox(String name, String message)

Displays an error message under the specified form field (either a control object or the name of the field). If the control or field is currently off the screen, the form scrolls to the control or field.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is off screen (scrolls the form to the control or field). The showFieldMsg() method is a similar method that requires a type parameter.

Parameters

Name	Type	Description
name	String	The name of the control or field.
message	String	The message to be displayed.

Returns

Type	Description
void	

GlideForm - showRelatedList(String listTableName)

Displays the specified related list on the form.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
listTableName	String	Name of the related list.

Returns

Type	Description
void	

GlideForm - showRelatedLists()

Displays all the form's related lists.

This method is not available on the mobile platform. If this method is run on a mobile platform, no action occurs.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - submit()

Saves the record.

The user is taken away from the form, returning them to where they were.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideForm - submit(String verb)

Performs the UI action specified by the parameter.

Parameters

Name	Type	Description
verb	String	An action_name from a sys_ui_action record. The action name must

Name	Type	Description
		be for a visible form button.

Returns

Type	Description
void	

Mobile GlideForm (g_form) - Client

Mobile GlideForm (g_form) methods enable you to work with forms on the mobile platform.

Use these methods in any script that targets a mobile device.

MobileGlideForm (g_form) - addDecoration(String fieldName, String icon, String text)

Adds a decorative icon next to a field.

Parameters

Name	Type	Description
fieldName	String	The field name.
icon	String	The font icon to show next to the field.
text	String	The text title for the icon (used for screen readers).

Returns

Type	Description
void	

Example

This example adds a VIP icon next to the caller.

```
function onChange(control, oldValue, newValue, isLoading)
{
      // if the caller_id field is not present, then we ca
n't add an icon anywhere
      if (!g_form.hasField('caller_id'))
          return;

      if (!newValue)
         return;

      g_form.getReference('caller_id', function(ref) {
      g_form.removeDecoration('caller_id', 'icon-star', 'V
IP');

      if (ref.getValue('vip') == 'true')
           g_form.addDecoration('caller_id', 'icon-star',
'VIP');
      });
      }
```

MobileGlideForm (g_form) - getLabel(String fieldName)

Gets the form label text.

Parameters

Name	Type	Description
fieldName	String	The field name.

Returns

Type	Description
String	The label text.

Example

```
if (g_user.hasRole('itil')) {
      var oldLabel = g_form.getLabel('comments');
      g_form.setLabel('comments', oldLabel + ' (Customer v
isible)');
      }
```

MobileGlideForm (g_form) - hasField(String fieldName)

Determines if a field is present on the form.

Present means that it can be shown, not that it is visible.

Parameters

Name	Type	Description
fieldName	String	The field to look for.

Returns

Type	Description
Boolean	True if the field is present on the form; false, if it is not. On the form means that the field is part of g_form. It could still be hidden, read-only, mandatory, or invalid.

Example

This example makes the assigned_to field mandatory if the assignment_group field is on the form.

```
if (g_form.hasField('assignment_group'))
      g_form.setMandatory('assigned_to', true);
```

MobileGlideForm (g_form) - removeDecoration(String fieldName, String icon, String text)

Removes a decorative icon from next to a field.

Parameters

Name	Type	Description
fieldName	String	The field name.
icon	String	The icon to remove.
text	String	The text title for the icon.

Returns

Type	Description
void	

Example

```
function onChange(control, oldValue, newValue, isLoading)
{
      // if the caller_id field is not present, then we ca
n't add an icon anywhere
      if (!g_form.hasField('caller_id'))
           return;

      if (!newValue)
           return;

      g_form.getReference('caller_id', function(ref) {
           g_form.removeDecoration('caller_id', 'icon-star
', 'VIP');

           if (ref.getValue('vip') == 'true')
                g_form.addDecoration('caller_id', 'icon-st
ar', 'VIP');
      });
      }
```

MobileGlideForm (g_form) - setLabel(String fieldName, String label)

Sets the form label text.

Parameters

Name	Type	Description
fieldName	String	The field name.
label	String	The field label text.

Returns

Type	Description
void	

Example

This example changes the comments label.

```
if (g_user.hasRole('itil')) {
      var oldLabel = g_form.getLabel('comments');
      g_form.setLabel('comments', oldLabel + ' (Customer v
isible)');
      }
```

GlideGuid - Client

You can create a globally unique identifier.

You access the GlideGuidV3 methods using the g_guid global object.

GlideGuid - generate(Number stringLength)

Creates a globally unique identifier 32 characters long, or as specified with the optional length argument.

Parameters

Name	Type	Description
stringLength	Number	The desired string length, must be between 1 and

Name	Type	Description
		32 inclusive. This parameter is optional. If not specified, the returned string will be 32 characters long.

Returns

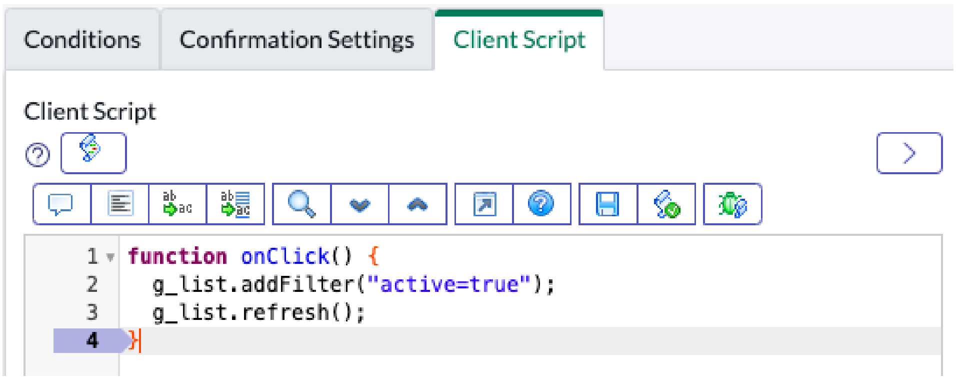
Type	Description
String	The globally unique identifier.

GlideList (Now Experience) - Client

Use the GlideList API to customize lists in the Next Experience UI Framework.

This API enables adding GlideList functionality to a button in the Workspace Experience UI. Use the `g_list` variable to call each method.

To add the button, create a list action and add GlideList method calls into the **Client Script** field `onClick{}` method. For instructions, see Use the client GlideList API in the Workspace Experience UI.



```
function onClick() {
  g_list.addFilter("active=true");
  g_list.refresh();
}
```

Note: Each method must be used with the refresh() method, with the exceptions of sort() and sortDescending().

The methods in this API are based on GlideList2. The following GlideList2 methods are not supported in the Next Experience framework and cause an error in the console log:

- isUserList()
- setFirstRow()
- showHideGroups()
- showHideList()
- toggleList()
- toggleListNoPref()

GlideList (Now Experience) - addFilter(String filter)

Adds a single term to the list query filter.

See also setFilter().

Parameters

Name	Type	Description
filter	String	Encoded query string in standard Glide format. See Encoded query strings.

Returns

Type	Description
None	

Example

The following example shows how to retrieve a list of active records and refresh the page.

```
g_list.addFilter("active=true");
g_list.refresh();
```

GlideList (Now Experience) - getChecked()

Returns a comma-separated list of the sys_ids for the items that are checked in the associated list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Comma-separated list of the sys_ids for the items that are checked in the list. Does not check to determine that the items returned are allowed to be executed.

Example

The following example shows how to get the sys_ids of each record selected in a list.

```
var myChecked = g_list.getChecked();
alert(myChecked);
g_list.refresh();
```

Output in an alert box (empty if no records are selected):

```
079893e6b733330059128ac7ee11a967, 4fca6d45b7131010f03e9b7a
de11a9d3, 5c460fbf1bd0011079e52131604bcbd9
```

GlideList (Now Experience) - getFixedQuery()

Returns the fixed query.

A fixed query is the part of the query that cannot be removed from the breadcrumb (i.e., it is fixed for the user).

Apply this method under **Related List Actions** in the Workspace Experience UI. For instructions, see Use the client GlideList API in the Workspace Experience UI.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The fixed query string for the list.

Example

The following example shows how to display a fixed query with a debug message in the browser console log.

```
console.log(g_list.getFixedQuery() + " debug message");
```

The output after clicking the button on the Child Incidents tab in an Incident record.

```
parent_incident=46f1784ba9fe19810018aa27fbb23482 debug mes
sage
```

GlideList (Now Experience) - getGroupBy()

Returns the field or comma-separated list of fields that are used to group the list.

See also:

- getQuery()
- setGroupBy()

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The field or comma-separated list of fields that are used to group the list.

Example

The following example shows how to get field groupBy values of the current list view.

```
var grpdBy = g_list.getGroupBy();
alert("The list is grouped by " + grpdBy);
g_list.refresh();
```

Output in an alert box for a list grouped by priority:

```
The list is grouped by ^GROUPBYpriority
```

GlideList (Now Experience) - getListName()

Returns the name of the list, which is usually the table name.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The list name (usually the table name).

Example

The following example shows how to call the method.

```
g_list.getListName();
```

GlideList (Now Experience) - getOrderBy()

Returns the first field used to order the list.

See also:

- getQuery()
- setOrderBy()

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The field by which to order the list. Empty if the list is not ordered.

Example

The following example shows how to get field orderBy values of the current list view.

```
var orderBy = g_list.getOrderBy();

alert("The list is ordered by " + orderBy);
g_list.refresh();
```

Output in an alert box for a list ordered by priority:

```
The list is ordered by ^ORDERBYDESCpriority
```

GlideList (Now Experience) - getParentTable()

Returns the name of the parent table for a related list (the table associated with the form).

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The parent table name.

Example

The following example shows how to call the method.

```
g_list.getParentTable();
g_list.refresh();
```

GlideList (Now Experience) - getQuery(Object)

Returns the encoded query string for the list.

See also:

- getFixedQuery()
- getGroupBy()

- getOrderBy()
- setGroupBy()
- setOrderBy()

Parameters

Name	Type	Description
<object>	Object	Optional. By default, this method includes orderBy, groupBy, and fixed query in the results. You can set object properties to restrict results to one or more of the three available options.
<object>.orderBy	Boolean	Optional. Flag that indicates whether to include orderBy in results. Valid values: • true: Include orderBy in results. • false: Do not include orderBy in results. Default: false
<object>.groupBy	Boolean	Optional. Flag that indicates whether to include groupBy in results. Valid values: • true: Include groupBy in results.

Name	Type	Description
		• false: Do not include groupBy in results. Default: false
<object>.fixed	Boolean	Optional. Flag that indicates whether to include fixed query in results. Valid values: • true: Include fixed query in results. • false: Do not include fixed query in results. Default: false

Returns

Type	Description
String	Encoded query string for the list.

Example

The following example shows how to display the list encoded query with a debug message in the browser console log.

```
var myQuery = g_list.getQuery();
console.log(myQuery + " message");
```

Output:

```
active=false^EQ message
```

GlideList (Now Experience) - getRelated()

Returns the related list field that associates the related list to the parent form.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Field that connects the list to the parent form.

Example

The following example shows how to call the method.

```
g_list.getRelated();
```

GlideList (Now Experience) - getTableName()

Returns the table name for the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Returns the table name for the list.

Example

The following example shows how to display the list name for the current view.

```
var listName = g_list.getListName();
alert("The current list name is " + listName);
g_list.refresh();
```

Output in an alert box:

```
The current list name is incident
```

GlideList (Now Experience) - getTitle()

Returns the list title.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The list title.

Example

The following example shows how to call the method.

```
g_list.getTitle();
```

GlideList (Now Experience) - getView()

Returns the view used to display the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The name of the view.

Example

The following example shows how to call the method.

```
g_list.getView();
```

GlideList (Now Experience) - refresh(Number firstRow)

Refreshes the list. The orderBy part of the list filter is ignored so that the list uses its natural ordering when it is refreshed.

Parameters

Name	Type	Description
firstRow	Number	The first row to appear in the list. Default: First row of the current view.

Returns

Type	Description
None	

Example

The following example shows how to call the method.

```
g_list.addFilter("active=true");
g_list.refresh();
```

GlideList (Now Experience) - refreshWithOrderBy(Number firstRow)

Refreshes the list. The orderBy part of the list filter is included if it is specified for the list.

Parameters

Name	Type	Description
firstRow	Number	The first row to appear in the list. Default: First row of the current view.

Returns

Type	Description
None	

Example

The following example shows how to call the method.

```
g_list.refreshWithOrderBy();
```

GlideList (Now Experience) - setFilter(String filter)

Sets the encoded query string for the list, ignoring the orderBy and groupBy parts of the query string.

See also addFilter().

Parameters

Name	Type	Description
filter	String	Encoded query string in standard Glide format. See Encoded query strings.

Returns

Type	Description
None	

Example

The following example shows how to restrict list results to active records.

```
g_list.setFilter("active=true");
g_list.refresh();
```

GlideList (Now Experience) - setGroupBy(String groupBy)

Sets the list groupBy criteria for a single field.

See also:

- getGroupBy()
- getQuery()

Parameters

Name	Type	Description
groupBy	String	The groupBy criteria for the list.

Returns

Type	Description
None	

Example

The following example shows how to group listed records by Caller ID.

```
g_list.setGroupBy("caller_id");
g_list.refresh();
```

GlideList (Now Experience) - setOrderBy(String orderBy)

Sets the orderBy criteria for the list.

For a single order by field, use orderBy field or orderByDesc field. For multiple fields, use `orderByField1^orderByField2^orderByField3`. orderBy specifies ascending order and orderByDesc specifies descending. These prefix strings are optional. If not specified, orderBy is the default order.

See also:

- getOrderBy()
- getQuery()
- sort()
- sortDescending()

Parameters

Name	Type	Description
orderBy	String	Single or multiple orderBy fields.

Returns

Type	Description
None	

Example

The following example shows how to order the list by the value of the Priority field.

```
g_list.setOrderBy("priority");
g_list.refresh();
```

GlideList (Now Experience) - setRowsPerPage(Number rows)

Sets the number of rows per page to display.

Parameters

Name	Type	Description
rows	Number	The number of rows to display.

Returns

Type	Description
None	

Example

The following example shows how to limit results to 50 rows per page.

```
g_list.setRowsPerPage(50);
refresh();
```

GlideList (Now Experience) - sort(String field)

Sorts the list in ascending order and sets the field as an orderBy column.

See also:

- getOrderBy()
- getQuery()
- setOrderBy()
- sortDescending()

Note: This method does not require the refresh() method.

Parameters

Name	Type	Description
field	String	Field to use to sort the list.

Returns

Type	Description
None	

Example

The following example shows how to sort results by the **Number** field.

```
g_list.sort("number");
```

GlideList (Now Experience) - sortDescending(String field, Number amount)

Sorts a single field in the list in descending order and sets the field as an orderByDescField column.

See also:

- getOrderBy()
- getQuery()

- setOrderBy()
- sortDescending()

Note: This method does not require the refresh() method.

Parameters

Name	Type	Description
field	String	Field to use to sort the list.

Returns

Type	Description
None	

Example

The following example shows how to sort results by the Number field in descending order.

```
g_list.sortDescending("number");
```

GlideList2 (g_list) - Client

GlideList2 is a JavaScript class used to customize (v2) lists.

The variable `g_list` is used to access a specified list object. The `g_list` variable is not available to the related lists form link UI action. It is available to the lists form link UI action.

These methods are used in UI Context Menus and UI Actions.

Several of these methods are available in Next Experience UI Framework. For details, see GlideList (Next Experience UI Framework).

GlideList2 - addFilter(String filter)

Adds a single term to the list query filter.

Parameters

Name	Type	Description
filter	String	Encoded query string in standard Glide format. See Encoded query strings.

Returns

Type	Description
void	

Example

```
g_list.addFilter("active=true");
```

GlideList2 - get(Object DOMelement)

Returns the GlideList2 object for the list that contains the specified item.

Parameters

Name	Type	Description
DOMelement	Object	The DOM element ID for the list for which you want the GlideList2 object.

Returns

Type	Description
Object	The GlideList2 object or null if not found.

GlideList2 - get(String ListID)

Returns the GlideList2 object for the list specified.

Parameters

Name	Type	Description
ListID	String	The list ID for which you want the GlideList2 object.

Returns

Type	Description
Object	The GlideList2 object or null if not found.

Example

```
function assignLabelActionViaLookupModal(tableName, listId
) {
        var list = GlideList2.get(listId);
        if (!list)
                return;

        assignLabelViaLookup(tableName, sysIds, list.getVi
ew());
}
```

GlideList2 - getChecked()

Returns a comma-separated list of the sys_ids for the items that are checked in the associated list.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	Comma-separated list of the sys_ids for the items that are checked in the list. Does not check to determine that the items returned are allowed to be executed.

Example

```
function removeLabelActionViaLookupModal(tableName, listId
) {
  var list = GlideList2.get(listId);
  if (!list)
    return;

  var sysIds = list.getChecked();
  if (!sysIds)
    return;

  removeLabelViaLookup(tableName, sysIds);
}
```

GlideList2 - getFixedQuery()

Returns the fixed query.

A fixed query is the part of the query that cannot be removed from the breadcrumb (i.e., it is fixed for the user). It is specified by including a sysparm_fixed_query parameter for the application module.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The fixed query string for the list.

Example

```
var list = GlideList2.get(container.readAttribute('list_id
'));
var filter = this._getFilter(element);
var fixedQuery = list.getFixedQuery();
if (fixedQuery)
  filter = fixedQuery + "^" + filter;
```

GlideList2 - getGroupBy()

Returns the field or comma-separated list of fields that are used to group the list.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The field or comma-separated list of fields that are used to group the list.

Example

```
function runFilterV2Lists(name, filter) {
  var list = GlideList2.get(name);
    if (list) {
      var groupBy = list.getGroupBy();
      if (groupBy)
        filter += "^" + groupBy;
        list.setFilterAndRefresh(filter);
    }
}
```

GlideList2 - getListName()

Returns the name of the list, which is usually the table name.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The list name (usually the table name).

Example

```
var list = GlideList2.get(name);
var listName = list.getListName();
```

GlideList2 - getOrderBy()

Returns the first field used to order the list.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The field by which to order the list. Empty if the list is not ordered.

Example

```
var list = GlideList2.get(listId);
if (!list)
  return;
var orderBy = list.getOrderBy();
```

GlideList2 - getParentTable()

Returns the name of the parent table for a related list (the table associated with the form).

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The parent table name.

Example

```
for (var id in GlideLists2) {
  var list = GlideLists2[id];
  if (list.getTableName() == listTableName && list.getPare
ntTable() == tableName)
    return list.getContainer();
}
```

GlideList2 - getQuery(Boolean orderBy, Boolean groupBy, Boolean fixed, Boolean all)

Returns the encoded query string for the list.

Parameters

Name	Type	Description
orderBy	Boolean	Optional. Flag that indicates whether to include orderBy in results. Valid values: • true: Include orderBy in results. • false: Do not include orderBy in results. Default: false
groupBy	Boolean	Optional. Flag that indicates whether to include groupBy in results. Valid values: • true: Include groupBy in results. • false: Do not include groupBy in results. Default: false
fixed	Boolean	Optional. Flag that indicates whether to include fixed query in results. Valid values:

Name	Type	Description
		• true: Include fixed query in results. • false: Do not include fixed query in results. Default: false
all	Boolean	Default. Flag that indicates whether to include orderBy, groupBy, and fixed query in results. Valid values: • true: Include orderBy, groupBy, and fixed query in results. • false: Do not include all three options in results. Default: true

Returns

Type	Description
String	Encoded query string for the list.

Example

```
var list = GlideList2.get(this.listID);
var ajax = new GlideAjax("AJAXJellyRunner", "AJAXJellyRunn
er.do");
  ajax.addParam("sysparm_query_encoded", list.getQuery({gr
oupby: true, orderby: true}));
```

```
  ajax.addParam("sysparm_table", list.getTableName());
  ajax.addParam("sysparm_view", list.getView());
```

GlideList2 - getRelated()

Returns the related list field that associates the related list to the parent form.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	Field that connects the list to the parent form.

Example

```
var list = GlideList2.get(name);
var related = list.getRelated();
if (related)
  ajax.addParam("sysparm_is_related_list", "true");
```

GlideList2 - getTableName()

Returns the table name for the list.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	Returns the table name for the list.

Example

```
GlideList2.getListsForTable = function(table) {
    var lists = [];
    for (var id in GlideLists2) {
        var list = GlideLists2[id];
        if (list.getTableName() == table)
            lists.push(list);
    }
    return lists;
}
```

GlideList2 - getView()

Returns the view used to display the list.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The name of the view.

Example

```
function assignLabelActionViaLookupModal(tableName, listId
) {
        var list = GlideList2.get(listId);
        if (!list)
                return;
        assignLabelViaLookup(tableName, sysIds, list.getVi
ew());
}
```

GlideList2 - getTitle()

Returns the list title.

Parameters

Name	Type	Description
none		

Returns

Type	Description
String	The list title.

Example

```
var list = GlideList2.get(name);
var listTitle = list.getTitle();
```

GlideList2 - isUserList()

Returns **true** if the list has been personalized by the user by choosing the list mechanic and changing the list layout.

Parameters

Name	Type	Description
none		

Returns

Type	Description
Boolean	True if the list layout has been changed.

Example

```
var list = GlideList2.get(listId);
if (!list)
  return;
if (list.isUserList())
  var tableName = list.getTableName();
```

GlideList2 - refresh(Number firstRow, String additionalParms)

Refreshes the list. The orderBy part of the list filter is ignored so that the list uses its natural ordering when it is refreshed.

Parameters

Name	Type	Description
firstRow	Number	The first row to appear in the list. Default: First row of the current view.
additionalParms	String	Optional name-value pairs that are submitted with the list refresh request.

Returns

Type	Description
void	

Example

```
$timeout(function() {
  if (GlideList.lists) {
    var list = GlideList.get(name);
    if (list) {
       if (sortBy) {
         if (sortDirection == 'ASC')
```

```
            list.sort(sortBy);
         else
            list.sortDescending(sortBy);
         }
       list.refresh();
     }
   }
}
```

GlideList2 - refreshWithOrderBy(Number firstRow, String description)

Refreshes the list. The orderBy part of the list filter is included if it is specified for the list.

Parameters

Name	Type	Description
firstRow	Number	The first row to appear in the list. Default: First row of the current view.
description	String	Optional name-value pairs that are submitted with the list refresh request.

Returns

Type	Description
void	

Example

```
ga.getXML(function(serverResponse) {
  var response = serverResponse.responseXML.getElementsByT
agName("response")[0];
  if (response) {
    var list = GlideList2.getByName("backlog_stories");
    list.refreshWithOrderBy();
    var status = response.getAttribute('status');
    $j('html, body').animate({scrollTop: $j("#"+data.recor
d.sys_id).offset().top},500);
    if (status == 'failure') {
      alert('${gs.getMessage("Story cannot be created. Tea
m is not associated with any project.")}');
    }
  }
}
```

GlideList2 - setFilter(String filter)

Sets the encoded query string for the list, ignoring the orderBy and groupBy parts of the query string.

Parameters

Name	Type	Description
filter	String	Encoded query string in standard Glide format. See Encoded query strings.

Returns

Type	Description
void	

Example

```
list = GlideList2.get($(side+"ContentDivRelease").select("
.list_div")[0].getAttribute("id"));
if (list) {
  list.setFilter("active=true");
  list.refresh(1);
 }
```

GlideList2 - setFilterAndRefresh(String filter)

Sets the encoded query string for the list, including the orderBy and groupBy if specified, and then refreshes the list using the new filter.

Parameters

Name	Type	Description
filter	String	Encoded query string.

Returns

Type	Description
void	

Example

```
function updateListFilter(projectID) {
  var list = GlideList2.getByName("backlog_stories");
  var fixedQuery = $('hdn_additional_filters').value;
  if(!projectID) {
      list.setFilterAndRefresh(fixedQuery + "^ORDERBYteam_
index");
      list.setOrderBy("team_index");
  }
}
```

GlideList2 - setFirstRow(Number rowNum)

Sets the first row that appears in the list when the list is refreshed.

Parameters

Name	Type	Description
rowNum	Number	Row number of the first row to display.

Returns

Type	Description
void	

Example

```
var nextRow = 0;
var rowsPerPage = 20;
var list = GlideList2.get(listId);
if (!list)
  return;
list.setFirstRow(nextRow);
nextRow = nextRow + rowsPerPage;
```

GlideList2 - setGroupBy(String groupBy)

Sets the list groupBy criteria for a single field.

Parameters

Name	Type	Description
groupBy	String	The groupBy criteria for the list.

Returns

Type	Description
void	

Example

```
function runContextAction(listId) {
  var g_list = GlideList2.get(listId);
  g_list.setGroupBy('');
  g_list.refresh(1);
}
```

GlideList2 - setOrderBy(String orderBy)

Sets the orderBy criteria for the list.

For a single order by field, use orderBy field or orderByDesc field.
For multiple fields, use `orderByField1^orderByField2^orderByField3`. orderBy specifies ascending order and orderByDesc specifies descending. These prefix strings are optional. If not specified, orderBy is the default order.

Parameters

Name	Type	Description
orderBy	String	Single or multiple orderBy fields.

Returns

Type	Description
void	

Example

```
                  updateOrderBy: function(orderBy){
  var list = GlideList2.get(this.listID);
  if (list)
    list.setOrderBy(orderBy);
};
```

GlideList2 - setRowsPerPage(Number rows)

Sets the number of rows per page to display.

Parameters

Name	Type	Description
rows	Number	The number of rows to display.

Returns

Type	Description
void	

Example

```
link: function(scope) {
  var list = GlideList2.get(scope.listId);
  list.setRowsPerPage(scope.maxRows);
  list.setFilterAndRefresh(scope.tableQuery);
}
```

GlideList2 - showHideGroups(Boolean showFlag)

Shows or hides all the groups within the list and saves the current collapsed/expanded state of the groups as a user preference.

Parameters

Name	Type	Description
showFlag	Boolean	If **true**, shows the groups within the list.

Returns

Type	Description
void	

Example

```
function showHideAllGroups(showFlag) {
  var list = GlideList2.get(listId);
  if (!list)
    return;
  list.showHideGroups(showFlag);
}
```

GlideList2 - showHideList(Boolean showFlag)

Displays or hides the list and saves the current collapsed/expanded state of the list as a user preference.

Parameters

Name	Type	Description
showFlag	Boolean	If **true**, displays the list.

Returns

Type	Description
void	

Example

```
GlideList2.toggleAll = function(expandFlag) {
for (var id in GlideLists2) {
  var list = GlideLists2[id];
list.showHideList(expandFlag);
}
```

GlideList2 - sort(String field)

Sorts the list in ascending order and sets the field as an orderBy column.

Parameters

Name	Type	Description
field	String	Field to use to sort the list.

Returns

Type	Description
void	

Example

```
$timeout(function() {
  if (GlideList.lists) {
    var list = GlideList.get(name);
    if (list) {
       if (sortBy) {
         if (sortDirection == 'ASC')
            list.sort(sortBy);
         else
            list.sortDescending(sortBy);
         }
       list.refresh();
     }
   }
}
```

GlideList2 - sortDescending(String field, Number amount)

Sorts a single field in the list in descending order and sets the field as an orderByDescField column.

Parameters

Name	Type	Description
field	String	Field to use to sort the list.

Returns

Type	Description
void	

Example

```
$timeout(function() {
  if (GlideList.lists) {
    var list = GlideList.get(name);
    if (list) {
       if (sortBy) {
         if (sortDirection == 'ASC')
            list.sort(sortBy);
         else
            list.sortDescending(sortBy);
         }
       list.refresh();
     }
   }
}
```

GlideList2 - toggleList()

Toggles the display of the list and saves the current collapsed/expanded state of the list as a user preference.

Parameters

Name	Type	Description
none		

Returns

Type	Description
void	

Example

```
var list = GlideList2.get(listId);
if (!list)
  return;
list.toggleList();
```

GlideList2 - toggleListNoPref()

Toggles the display of the list but does not save the current collapsed/expanded state of the list as a user preference.

Parameters

Name	Type	Description
none		

Returns

Type	Description
void	

Example

```
var list = GlideList2.get(listId);
if (!list)
  return;
list.toggleListNoPref();
```

GlideListV3 (g_list) - Client

Use GlideListV3 to manipulate lists.

You access the GlideListV3 methods by using the g_list global object. These methods are used in UI context menus and UI actions. The g_list object is not available for related lists on the form link UI action.

Note:

This API is no longer supported. Consider using the GlideList2() API instead.

GlideListV3 - addFilter(String filter)

Adds a single term to the list query filter.

Parameters

Name	Type	Description
filter	String	Query string condition to add.

Returns

Type	Description
void	

GlideListV3 - get(String listId)

Returns the GlideList object for specified list.

Parameters

Name	Type	Description
listId	String	The list name.

Returns

Type	Description
Object	The GlideList object for the specified list, or null if not found.

GlideListV3 - get(Object DomElement)

Returns the GlideList object for the specified DOM element.

Parameters

Name	Type	Description
DomElement	Object	The DOM element ID for which you want the GlideList object.

Returns

Type	Description
Object	The GlideList object for the specified DOM element. Returns null if the DOM element is not found.

GlideListV3 - getChecked()

Returns a comma-separated list of sys_ids for checked items in the list. Does not return items that are not allowed to be executed.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Comma-separated list of the sys_ids for checked items in the list. Does not return items that are not allowed to be executed.

GlideListV3 - getFixedQuery()

Returns the sysparm_fixed query.

The fixed query is the part of the query that cannot be removed from the breadcrumb (i.e., it is fixed for the user). It is specified by including a sysparm_fixed_query parameter for the application module.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The fixed query string for the list.

GlideListV3 - getFormTarget()

Returns the form's target attribute.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The form's target attribute.

GlideListV3 - getGroupBy()

Returns the field or comma-separated list of fields that are used to group the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The field or comma-separated list of fields used to group the list.

GlideListV3 - getListName()

Returns the name of the list, which is usually the table name.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The list name.

GlideListV3 - getOrderBy()

Returns the first field used to order the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The field used to order the list, or an empty string if the list is not sorted.

GlideListV3 - getParentTable()

Returns the name of the parent table (the table associated with the form).

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The parent table name.

GlideListV3 - getQuery(Object options)

Returns the encoded query string for the list.

Parameters

Name	Type	Description
options	Object	The options can be one or more of the following. • orderby - include ORDERBY in the query • groupby - include GROUPBY in the query • fixed - include sysparm_fixed_query in the query • all - include all the options in the query

Returns

Type	Description
String	Encoded query string for the list.

GlideListV3 - getReferringUrl()

Returns the referring URL.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Returns the parent form's URL, or '*' if there is no parent form.

GlideListV3 - getRelated()

Returns the related list field that associates the related list to the parent form.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Field that connects the list to the parent form.

GlideListV3 - getRelatedListType()

Returns the related list type.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The relationship table type.

GlideListV3 - getRelationshipId()

Returns the relationship record id, if this is type REL related list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The sys_id of the relationship record.

GlideListV3 - getRowCount()

Returns the number of rows returned by the query.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Number	The number of rows returned by the query.

GlideListV3 - getRowsPerPage()

Returns the number of rows to be displayed on a page.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Number	The number of rows to be displayed on a page.

GlideListV3 - getTableName()

Returns the table name of the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The list's table name.

GlideListV3 - getTitle()

Returns the list title.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The list title.

GlideListV3 - getView()

Returns the view used to display the list.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The name of the view

GlideListV3 - isUserList()

Returns true if the list has been personalized by the user.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	True if the list layout has changed.

GlideListV3 - refresh(Number firstRow, Object additionalParams)

Refreshes the list. The orderBy part of the list filter is ignored so that the list's natural ordering is used.

Parameters

Name	Type	Description
firstRow	Number	(Optional) The first row to display in the list. If not specified, the list's current first row is used.
additionalParams	Object	(Optional) Name-value pairs that are submitted with the list refresh request.

Returns

Type	Description
void	

GlideListV3 - refreshWithOrderBy(Number firstRow, Object additionalParams)

Refreshes the list using the orderBy fields.

Parameters

Name	Type	Description
firstRow	Number	(Optional) The first row to display in the list. If not specified, the list's current first row is used.
additionalParams	Object	(Optional) Name-value pairs that are submitted with the list refresh request.

Returns

Type	Description
void	

GlideListV3 - setFilter(String filter, Boolean saveOrderBy, Boolean saveGroupBy)

Sets the encoded query string for the list ignoring the orderBy and groupBy parts of the query string.

Parameters

Name	Type	Description
filter	String	An encoded query string.
saveOrderBy	Boolean	The default is false. When true uses the orderBy part of the query.
saveGroupBy	Boolean	The default is false. When true uses the

Name	Type	Description
		groupBy part of the query.

Returns

Type	Description
void	

GlideListV3 - setFilterAndRefresh(String filter)

Sets the encoded query string for the list, and then refreshes the list using the new filter.

This preserves the groupby and orderby parameters.

Parameters

Name	Type	Description
filter	String	Encoded query string.

Returns

Type	Description
void	

GlideListV3 - setFirstRow(Number firstRow)

Sets the first row to be displayed when the list is refreshed.

Parameters

Name	Type	Description
firstRow	Number	The row number in the list.

Returns

Type	Description
void	

GlideListV3 - setFormTarget(String target)

Specifies where to display the response from the form.

Parameters

Name	Type	Description
target	String	The form.target attribute value to use.

Returns

Type	Description
void	

GlideListV3 - setGroupBy(String groupBy)

Sets the groupBy criteria for the list, for a single field or multiple fields.

For a single field, use field or groupByField. The groupBy prefix is optional. For multiple fields use `field1^field2^field3` or `groupByField1^groupByField2^groupByField3`.

Parameters

Name	Type	Description
String	groupBy	The group by criteria for the list.

Returns

Type	Description
void	

GlideListV3 - setOrderBy(String orderBy)

Sets the orderBy criteria for the list.

For a single order by field use orderBy field or orderByDescField. For multiple fields, use `orderByField1^orderByField2^orderByField3`. orderBy specifies ascending order and orderByDesc specifies descending. These prefix strings are optional. If not specified orderBy is assumed.

Parameters

Name	Type	Description
orderBy	String	Single or multiple order by fields.

Returns

Type	Description
void	

GlideListV3 - setReferringUrl(String url)

Sets the parent form referring url.

Parameters

Name	Type	Description
url	String	The parent form's URL

Returns

Type	Description
void	

GlideListV3 - setRowsPerPage(Number numRows)

Set the number of rows to display on a page.

Parameters

Name	Type	Description
numRows	Number	The number of rows to display on a page.

Returns

Type	Description
void	

GlideListV3 - showHideGroups(Boolean showFlag)

Displays or hides all of the groups within the list and saves the current collapsed/expanded state of the groups as a user preference.

Parameters

Name	Type	Description
showFlag	Boolean	When true, displays the groups within the list.

Returns

Type	Description
void	

GlideListV3 - showHideList(Boolean showFlag)

Displays or hides the list and saves the current collapsed/expanded state of the list as a user preference.

Parameters

Name	Type	Description
showFlag	Boolean	When true, displays the list.

Returns

Type	Description
void	

GlideListV3 - sort(String field)

Sort the list in ascending order.

Parameters

Name	Type	Description
field	String	The field to be used to sort the list.

Returns

Type	Description
void	

GlideListV3 - sortDescending(String field)

Sorts the list in descending order.

Parameters

Name	Type	Description
field	String	The field used to sort the list.

Returns

Type	Description
void	

GlideListV3 - toggleList()

Toggles the list display between collapsed and expanded, and saves the state as a user preference.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideListV3 - toggleListNoPref()

Toggles the list display between collapsed and expanded, but does not save the state as a user preference.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideMenu (g_menu and g_item) - Client

GlideMenu methods are used in UI Context Menus, in the onShow scripts to customize UI Context Menu items.

There is no constructor for the GlideMenu class. Access GlideMenu methods using the g_menu global object.

- g_menu is the UI Context Menu that is about to be shown. The onShow script can make changes to the appearance of the menu before it is displayed using these methods.
- g_item is the current UI Context Menu item that is about to be shown. It is used in several of the g_menu methods to specify an item.

GlideMenu - clearImage(GlideMenuItem item)

Clears the image for an item.

Parameters

Name	Type	Description
item	GlideMenuItem	Specifies the item to have its image removed from display.

Returns

Type	Description
void	

Example

```
g_menu.clearImage(g_item);
```

GlideMenu - clearSelected()

Clears any selection images from items in the menu.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

GlideMenu - getItem(String itemID)

Returns a menu item by item ID.

It can be necessary to find an item in a menu so that it can be changed before being displayed. Each menu item may be assigned a unique ID when the menu item is created (either from a UI Context Menu entry or from the addAction() method in the Dynamic Script Action).

Parameters

Name	Type	Description
itemID	String	Specifies the item to be returned.

Returns

Type	Description
GlideMenuItem	The menu item

GlideMenu - setDisabled(GlideMenuItem item)

Disables a menu item so that it cannot be selected. The disabled menu item is displayed in a lighter color (grayed out) to indicate it is disabled.

Parameters

Name	Type	Description
item	GlideMenuItem	The item to be disabled.

Returns

Type	Description
void	

Example

```
g_menu.setDisabled(g_item);
```

GlideMenu - setEnabled(GlideMenuItem item)

Enables the specified menu item.

Parameters

Name	Type	Description
item	GlideMenuItem	The item to be enabled.

Returns

Type	Description
void	

Example

```
g_menu.setEnabled(g_item);
```

GlideMenu - setHidden(GlideMenuItem item)

Hides the specified menu item.

When hiding menu items, the separator bars are not adjusted, so it is possible to end up with the menu showing two separators in a row.

Parameters

Name	Type	Description
item	GlideMenuItem	The item to be hidden.

Returns

Type	Description
void	

Example

```
g_menu.setHidden(g_item);
```

GlideMenu - setImage(GlideMenuItem item, String imgSrc)

Sets an image for an item.

Parameters

Name	Type	Description
item	GlideMenuItem	the item to have the image displayed.
imgSrc	String	the image to attach to the menu item.

Returns

Type	Description
void	

Example

```
g_menu.setImage(g_item, 'images/checked.gifx');
```

GlideMenu - setLabel(GlideMenuItem item, String label)

Sets the display label for a menu item. The label may contain HTML.

Parameters

Name	Type	Description
item	GlideMenuItem	the item to be labeled.
label	String	the label to be displayed. The string may contain HTML.

Returns

Type	Description
void	

Example

```
g_menu.setLabel(g_item, "This is a new label");
```

GlideMenu - setVisible(GlideMenuItem item)

Displays the specified item.

Parameters

Name	Type	Description
item	GlideMenuItem	The item to be displayed.

Returns

Type	Description
void	

Example

```
g_menu.setVisible(g_item);
```

GlideModalForm - Client

Displays a form in a GlideModal.

General usage of the GlideModalForm class involves creating the object, setting any preferences, and then rendering the GlideModalForm.

```
var d = new GlideModalForm('dialog title', 'table_name_or_
form_name', [callback on completion of submit])
         d.setPreference('name', 'value');
         d.render();
```

Specify the query parameters that are passed to the form using setPreference(). Any name/value pair that you specify with setPreference() is sent along with the form POST request to display the form.

The GlideModalForm is set to fill the height of the document window.

GlideModalForm - GlideModalForm(String title, String tableName, Function onCompletionCallback, Boolean readOnly)

Creates an instance of the GlideModalForm class.

Parameters

Name	Type	Description
title	String	Modal form title.
tableName	String	Table being shown.

Name	Type	Description
onCompletionCallback	Function	Function to call after the form has been submitted and processed on the server. The callback function has the form`callbackFunction(String action_verb, String sys_id, String table, String displayValue)` where: • action_verb: Name of the UI action executed. Examples are sysverb_insert (Submit button), sysverb_cancel, sysverb_save (Save button). • sys_id: Sys_id of the affected record. • table: Name of the table containing the record. • displayValue: Value that appears on the form.
readOnly	Boolean	Optional. Flag that indicates whether the modal form should be set to read only.

Name	Type	Description
		Valid values: • true: Set form to read only. • false: Set for to read/write. Default: false

Example

This example shows how to instantiate a GlideModalForm object.

```
function openDevice(deviceSysID, deviceName) {
  var uName = gel('hidden_user_name').value + "'s ";
  deviceName = new String(deviceName).escapeHTML();
  var gp = new GlideModalForm(uName + deviceName, "cmn_not
if_device", refreshNotifPage);
  gp.addParm('sys_id', deviceSysID);
  gp.render();
}
```

GlideModalForm - addParm(String name, String value)

Sets the specified form field to the specified value.

Parameters

Name	Type	Description
name	String	Form field name. If the specified name is not a field in the associated modal form, it is ignored.
value	String	Value to set the specified form field to.

Returns

Type	Description
void	

Example

This example shows how to call addParm() to set the value of the sys_id field the modal form.

```
function openDevice(deviceSysID, deviceName) {
  var uName = gel('hidden_user_name').value + "'s ";
  deviceName = new String(deviceName).escapeHTML();
  var gp = new GlideModalForm(uName + deviceName, "cmn_not
if_device", refreshNotifPage);
  gp.addParm('sys_id', deviceSysID);
  gp.render();
}
```

GlideModalForm - setSysID(String sys_id)

Sets the object's sys_id preference.

Parameters

Name	Type	Description
sys_id	String	The id preference. One of the query parameters passed to the form.

Returns

Type	Description
void	

Example

This example shows how to use the setSysID() method to initialize the value of the sys_id.

```
function(startDate, endDate) {
  var dialog = new GlideModalForm("Add Schedule Item", "cm
n_schedule_span");
  dialog.setSysID("-1");
  dialog.addParm("sysparm_collection", "cmn_schedule");
  dialog.addParm("sysparm_collectionID", this.sysId);
  dialog.addParm("sysparm_collection_key", "schedule");

  var q = "schedule=" + this.sysId + "^start_date_time="
   + startDate.serializeInUserFormat() + "^end_date_time="
   + endDate.serializeInUserFormat() + "^";

  if (startDate.isAllDay(endDate))
    q += "^all_day=true^";

  dialog.addParm("sysparm_query", q);
  dialog.render();
}
```

GlideModalForm - setCompletionCallback(Function callbackFunction)

Sets the function to be called when the form has been successfully submitted and processed by the server.

Parameters

Name	Type	Description
callbackFunction	Function	Callback function to call when the form has been successfully processed. The callback function has the form `callbackFunction(String action_verb, String sys_id,`

Name	Type	Description
		`String table, String displayValue)` where: • action_verb: action_name from a sys_ui_action record • sys_id: Sys_id of the affected record • table: Name of the table containing the record • displayValue: Value that appears on the form

Returns

Type	Description
void	

Example

This example shows how to set the onload callback function of the associated modal.

```
function handleCreateOrEdit(targetFieldName, sourceFieldNa
me, adapterRuleId, transformerSysId){
  dialog = new GlideModalForm('Edit Adapter Rule', "sys_ad
apter_rule");
  dialog.setSysID(adapterRuleId); //Pass in sys_id to edi
t existing record
  dialog.addParm('sysparm_form_only', 'true'); //Add or re
move related lists
  dialog.setOnloadCallback(hideModalForm);
  dialog.setCompletionCallback(handleAdapterCreatedOrUpdat
ed);
  dialog.render(); //Open the dialog
}
function handleAdapterCreatedOrUpdated(action_verb, sys_id
, table, displayValue) {
  var draftRecordTransformer = g_form.getValue("draft_reco
rd_transformer");
  if(draftRecordTransformer == null || draftRecordTransfor
mer.length == 0) {
    //sync Sticky Replications if it is enabled.
    var ajax = new GlideAjax('ReplicationPoolUtil');
    ajax.addParam('sysparm_name', 'syncStickyReplicationSe
t');
    ajax.addParam('sysparm_entry_set', g_form.getValue("en
try_set"));
    ajax.getXMLWait();
  }
}
```

GlideModalForm - setOnloadCallback(Function callbackFunction)

Sets the function to be called after the form has been loaded.

Parameters

Name	Type	Description
callbackFunction	Function	Function to call after the form has been loaded. The callback function has the form `callBackFunction(GlideModalForm obj)`

Returns

Type	Description
void	

Example

This example shows how to set the on load callback function of the associated modal.

```
function handleCreateOrEdit(targetFieldName, sourceFieldNa
me, adapterRuleId, transformerSysId){
  dialog = new GlideModalForm('Edit Adapter Rule', "sys_ad
apter_rule");
  dialog.setSysID(adapterRuleId); //Pass in sys_id to edi
t existing record
  dialog.addParm('sysparm_form_only', 'true'); //Add or re
move related lists
  dialog.setOnloadCallback(hideModalForm);
  dialog.setCompletionCallback(handleAdapterCreatedOrUpdat
ed);
  dialog.render(); //Open the dialog
}
```

GlideModalForm - render()

Shows the modal form.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

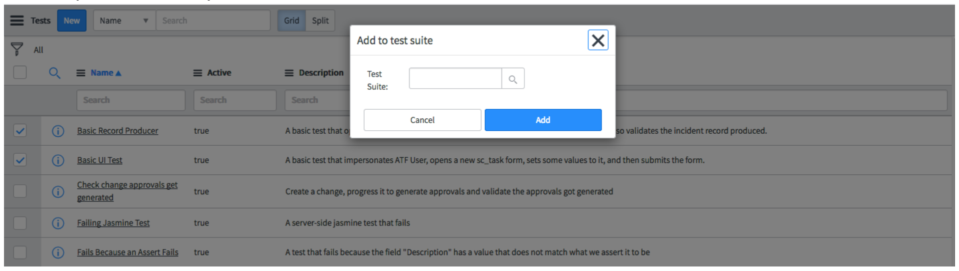
This example shows how to call render() to display the modal.

```
function openDevice(deviceSysID, deviceName) {
  var uName = gel('hidden_user_name').value + "'s ";
  deviceName = new String(deviceName).escapeHTML();
  var gp = new GlideModalForm(uName + deviceName, "cmn_not
if_device", refreshNotifPage);
  gp.addParm('sys_id', deviceSysID);
  gp.render();
}
```

GlideModal - Client

Provides methods for displaying a content overlay.

This is a fully-featured replacement for GlideWindow and GlideDialogWindow.
Example overlay



	Name ▲	Active	Description
✓	Basic Record Producer	true	A basic test that o so validates the incident record produced.
✓	Basic UI Test	true	A basic test that impersonates ATF User, opens a new sc_task form, sets some values to it, and then submits the form.
	Check change approvals get generated	true	Create a change, progress it to generate approvals and validate the approvals got generated
	Failing Jasmine Test	true	A server-side jasmine test that fails
	Fails Because an Assert Fails	true	A test that fails because the field "Description" has a value that does not match what we assert it to be

GlideModal - GlideModal(String id, Boolean readOnly, Number width)

Creates an instance of the GlideModalV3 class.

Parameters

Name	Type	Description
id	String	The UI page to load into the modal.
readOnly	Boolean	When true, hides the close button.
width	Number	The width in pixels.

GlideModal - get(String id)

Get a GlideModal object by ID.

Parameters

Name	Type	Description
id	String	The element id of the GlideModal object.

Returns

Type	Description
GlideModal	The object.

GlideModal - getPreference(String name)

Returns the value of the specified preference (property).

Invoking actions that create the modal typically also create the necessary preferences for the modal using the GlideModal - setPreference(String name, String value) method. The UI page client script then consumes these preferences using this method.

Parameters

Name	Type	Description
name	String	Name of the preference value to retrieve. This value must have previously been set on the modal using the GlideModal - setPreference(String name, String value) method.

Returns

Type	Description
String	Specified preference's value.

Example

This example shows a simple case of setting a preference and then retrieving that preference from a specified modal.

```
var gm = new GlideModal('UI_dialog_name');
//Sets the dialog title
gm.setTitle('Show title');

//sets the value of the preference table
gm.setPreference('table', 'incident');

//gets the value of the preference table
var title = gm.getPreference('table');
```

GlideModal - render()

Renders the UI page in the modal.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
var gm = new GlideModal("UI_dialog_name");

//Sets the dialog title
gm.setTitle('Show title');
gm.setWidth(550);

//Opens the dialog
gm.render();
```

GlideModalV3 - renderWithContent(Object html)

Display a modal with the specified HTML content.

The renderWithContent() method replaces the render() method, and does not request a UI page to render.

Parameters

Name	Type	Description
html	Object	The HTML content to be shown in the modal.

Returns

Type	Description
void	

GlideModal - renderWithContent(String html)

Display a modal with the specified HTML content.

The renderWithContent() method replaces the render() method, and does not request a UI page to render.

Parameters

Name	Type	Description
html	String	The HTML content to be shown in the modal.

Returns

Type	Description
void	

GlideModal - setPreference(String name, String value)

Sets the specified field on the current form to the specified value.

Parameters

Name	Type	Description
name	String	Name of the form field to update. If this field does not exist on the current form, the request is ignored.
value	String	Value to store in the specified form field.

Returns

Type	Description
void	

Example

```
var gm = new GlideModal('UI_dialog_name');
//Sets the dialog title
gm.setTitle('Show title');
gm.setPreference('table', 'task');
gm.setPreference('name', 'value');

//Opens the dialog
gm.render();
```

GlideModal - setPreferenceAndReload(Array properties)

Set the properties and reload the modal.

Parameters

Name	Type	Description
properties	Array	An array of name-value pairs to be set.

Returns

Type	Description
void	

GlideModal - setTitle(String title)

Sets the title of the modal.

Parameters

Name	Type	Description
title	String	The title to be displayed

Returns

Type	Description
void	

Example

```
var gm = new GlideModal('UI_dialog_name');

//Sets the dialog title
gm.setTitle('Show title');
gm.setPreference('name', 'value');

//Opens the dialog
gm.render();
```

GlideModal - setWidth(Number width)

Set the width in pixels.

The modal is boxed into predefined system sizes.

Parameters

Name	Type	Description
width	Number	The number of pixels.

Returns

Type	Description
void	

Example

```
var gm = new GlideModal('UI_dialog_name');

//Sets the dialog title
gm.setTitle('Show title');
gm.setPreference('name', 'value');

gm.setWidth(550);

//Opens the dialog
gm.render();
```

GlideModal - switchView(String newView)

Change the view and reload the modal.

Parameters

Name	Type	Description
newView	String	The view to use.

Returns

Type	Description
void	

GlideNavigation - Client

Provides methods to control and refresh the navigator and main frame.

The GlideNavigation methods are accessed using the g_navigation global object.

GlideNavigation - open(String url, String target)

Redirects to a new URL.

Parameters

Name	Type	Description
url	String	The URL to load. It can be any URL supported by the browser.
target	String	Optional. The frame in which to load the content specified by the URL. Default: Current frame

Returns

Type	Description
void	

Example

This example shows how to open the list of active incidents within an instance and display the content in the current frame.

```
g_navigation.open('incident_list.do?sysparm_query=active=t
rue');
```

GlideNavigation - openPopup(String url, String name, String features, Boolean noStack)

Opens the specified URL in a popup window.

The features parameter is part of the DOM specification and is passed through. For more information on the available feature list, refer to the Mozilla Developer Network.

Parameters

Name	Type	Description
url	String	URL to open.
name	String	Window name.
features	String	Comma separated list of features for the popup window.
noStack	Boolean	Flag that indicates whether to append `sysparm_stack=no` to the URL. This parameter helps prevent unexpected behavior when using the form back button. Valid values: • true: Append `sysparm_stack=no` to the URL.

Name	Type	Description
		• false: Do not append `sysparm_stack=no` to the URL.

Returns

Type	Description
Window	Instance of the new window.

Example

This example shows how to open the list of active incidents within a popup window called "Active Incidents", and enable the resizable, scrollbars, and status features on the window.

```
g_navigation.openPopup('incident_list.do?sysparm_query=active=true', 'Active Incidents', 'resizable,scrollbars,status', true);
```

GlideNavigation - openRecord(String tableName, String sys_id)

Redirects to a record. The record displays in the navigator frame.

Parameters

Name	Type	Description
tableName	String	Name of the table containing the record to display.
sys_id	String	Sys_id of the record to display.

Returns

Type	Description
void	

Example

This example shows how to open a specified incident record in the navigator frame.

```
g_navigation.openRecord('incident', '4e49c0e81bf198101363f
f37dc4bcb8a');
```

GlideNavigation - refreshNavigator()

Refreshes content in the navigator frame.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

This example shows how to refresh the content in the navigator frame.

```
g_navigation.refreshNavigator();
```

GlideNavigation - reloadWindow()

Reloads the current frame.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

This example shows how to refresh the content in the current frame.

```
g_navigation.reloadWindow();
```

GlideNotification - Client

You can show messages over the page content.

The GlideNotification method is accessed using the g_notification global object. List V3 must be activated for the g_notification object to be available.

GlideNotification - show(String type, String message)

Displays the specified string over the page content as the specified type of message.

Parameters

Name	Type	Description
type	String	Type of message to display. For example: • error • info

Name	Type	Description
		• warning
message	String	Message to display.

Returns

Type	Description
void	

Example

```
// Displays an info message at the top of the screen
nowapi.g_notification.show("info", "The record has been up
dated");

// Displays an error message at the top of the screen
nowapi.g_notification.show("error", "You need to provide n
otes!");
```

GlideRecord - Client

GlideRecord is used for database operations. The client-side GlideRecord API enables the use of some GlideRecord functionality in client-side scripts, such as client scripts and UI policy scripts.

A GlideRecord contains both records and fields. Queries made with the client-side GlideRecord are executed on the server. Therefore, a request is made from the client browser to obtain the record data.

The client-side GlideRecord API is not supported in scoped applications. Instead, create a script include and use the GlideAjax API, or use the REST APIs. In addition, the client-side GlideRecord API applies ACLs based on the credentials of the user executing the script. To execute the code on the server without ACLs, use the GlideAJAX API.

Client side GlideRecord - GlideRecord(String tableName)

Creates an instance of the GlideRecord class for the specified table.

Parameters

Name	Type	Description
tableName	String	The table to be used.

Example

```
var now_GR = new GlideRecord('incident');
```

Client side GlideRecord - addOrderBy(String column)

Adds a column to order by in the query.

Parameters

Name	Type	Description
column	String	The column by which to order the result set.

Returns

Type	Description
void	

Client side GlideRecord - addQuery(String name, Object value)

Adds a query to return records where the specified field name is equal to a specified value (or is in a list of values).

If you are familiar with SQL, this method is similar to the "where" clause. You can create one or more queries for a single filter by calling this method multiple times; for this method the queries are AND'ed. Once you define all of the desired queries, call the Client side GlideRecord - query(String name, Function responseFunction, String value) to execute the specified query clause (filter).

To perform an operation other than AND, use either the addQuery(String name, Object operator, Object value) method or the setEncodedQuery() method.

Parameters

Name	Type	Description
name	String	Name of the field to check.
value	Object	The value or list of values on which to query.

Returns

Type	Description
void	

Example

Example showing multiple queries.

```
var now_GR = new GlideRecord('incident');
now_GR.addQuery('priority', 4); // Priority is 4 - Low and
,
now_GR.addQuery('state', 3); // State is On Hold
now_GR.query(response);

function response(result) {
    while(result.next()) {
       // Print all INC with priority 4 - Low AND state i
s On Hold
        console.log(result.getValue('number'));
    }
}
```

Example

Example showing how to pass a string object.

```
var now_GR = new GlideRecord('incident');
now_GR.addQuery('short_description', new String('USB devic
e not working')); // string object
```

```
now_GR.addQuery('priority', 4); //number
now_GR.query(response);

function response(result) {
    while(result.next()) {
       // Print all INC with priority 4 - Low AND short de
scription contains 'USB device not working'
        console.log(result.getValue('number'));
    }
}
```

Client side GlideRecord - addQuery(String name, Object operator, Object value)

Adds a filter to return records where the field meets the specified condition (field, operator, value).

If you are familiar with SQL, this method is similar to the "where" clause. You can create one or more queries for a single filter by calling this method multiple times. Once you define all of the desired queries, call the Client side GlideRecord - query(String name, Function responseFunction, String value) to execute the specified query clause (filter).

To create more complex queries, use the setEncodedQuery() method.

Parameters

Name	Type	Description
name	String	Name of the field to check.
operator	Object	Query operator. The available values are dependent on the data type of the value parameter. Numbers: • = • !=

Name	Type	Description
		• > • >= • < • <= Strings (must be in upper case): • = • != • IN • NOT IN • STARTSWITH • ENDSWITH • CONTAINS • DOES NOT CONTAIN • INSTANCEOF **Note:** Use CONTAINS instead of the LIKE operator.
value	Object	Value on which to query (not case-sensitive).

Returns

Type	Description
void	

Example

Example showing how to add multiple queries to a filter.

```
var now_GR = new GlideRecord('incident');
now_GR.addQuery('priority', '<=', 2); // Priority is 2 or
higher and,
now_GR.addQuery('short_description', 'CONTAINS', 'crash')
; // Short description contains the word crash
now_GR.query(response);

function response(result) {
    while(result.next()) {
       // Print all INC with priority of 2 or higher AND s
hort description contains "crash"
        console.log(result.getValue('number'));
    }
}
```

Example

Example showing how to pass in an array to verify multiple conditions in a single query.

```
var priorities = [4,2];
var now_GR = new GlideRecord('incident');
now_GR.addQuery('priority', 'IN', priorities);
now_GR.query(response);

function response(result) {
  while(result.next()) {
    console.log(result.getValue('number'));
  }
}
```

Client side GlideRecord - deleteRecord(Function responseFunction)

Deletes the current record and calls the specified response function when complete.

Parameters

Name	Type	Description
responseFunction	Function	Response function for the callback.

Returns

Type	Description
None	

Example

This example deletes a record and then calls the response function response to log an alert message.

```
var recordGR = new GlideRecord('incident);
if (recordGR.get('99ebb4156fa831005be8883e6b3ee4b9')) {
  recordGR.deleteRecord(response);
}

function response(result) {
  alert('Deleted record sys_id: ' + result.getValue('sys_i
d'));
}
```

Output:

```
Deleted record sys_id: 99ebb4156fa831005be8883e6b3ee4b9
```

Client side GlideRecord - get(String sys_id)

Executes a GlideRecord query for a record with the specified sys_id. This method is expected to be used to query for single records, so a next operation is performed before returning.

Parameters

Name	Type	Description
sys_id	String	The sys_id of the record to be found.

Returns

Type	Description
Boolean	True if one or more matching records was found. False if no records were found.

Client side GlideRecord - getEncodedQuery()

Retrieves the query condition of the current result set as an encoded query string.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	An encoded query string containing all conditions that have been added to the query.

Client side GlideRecord - getLimit()

Returns the limit for records to be returned by the GlideRecord query.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Number	The limit for records to be returned by the GlideRecord query.

Client side GlideRecord - getTableName()

Retrieves the name of the table associated with this GlideRecord.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The table name

Example

```
var item = new GlideRecord('sc_request');
item.addQuery('sys_id', current.request);
item.query(itemResponse);

function itemResponse(item) {
   alert('The table is ' + item.getTableName());
}
```

Client side GlideRecord - hasNext()

Determines if there are any more records in the GlideRecord.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	True if there are more records in the query set.

Client side GlideRecord - insert(Function responseFunction)

Inserts a new record using the field values that have been set for the current record.

Parameters

Name	Type	Description
responseFunction	Function	Function to execute once the record is inserted.

Returns

Type	Description
String	Sys_id of the inserted record, or null if the record was not inserted.

Example

```
var now_GR = new GlideRecord('incident');
now_GR.short_description = 'Learn about GlideRecord';
var recResponse = now_GR.insert(handleResponse);
```

```
function handleResponse(recResponse, answer) {
// Answer will be the sys_id of the created record or null
alert('Newly created sys_id is - ' + answer + ' exists');
}
```

Client side GlideRecord - next()

Moves to the next record in the GlideRecord.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	False if there are no more records in the query set.

Example

```
var rec = new GlideRecord('incident');
rec.query(recResponse);

function recResponse(rec) {
  while (rec.next()) {
    alert(rec.number + ' exists');
  }
}
```

Client side GlideRecord - orderBy(String column)

Specifies an orderBy column. May be called more than once to order by multiple columns.

Parameters

Name	Type	Description
column	String	The column name to be used to order the result set.

Returns

Type	Description
void	

Example

```
function UpdateProjectWBS(project) {
  var count = 0;
  var child = new GlideRecord('pm_project_task');
  child.addQuery('parent', project.sys_id);
  child.orderBy('order');
  child.orderBy('number');
  child.query(childResponse);
  g_form.addInfoMessage(count + ' Project Tasks updated');
}

function childResponse(child) {
  var len = child.getRowCount().toString().length;
  var seq = 0;
  while (child.next()) {
    count += UpdateProjectTaskWBS(child, 1, ++seq, len, ''
);
  }
}
```

Client side GlideRecord - query(String name, Function responseFunction, String value)

Runs the query to the server against the table based on the addQuery() filter. This method queries the GlideRecord table as well as any references of the table.

Do not make synchronous query calls. Performing a query without a response function makes the call synchronous, which means that the display will wait for the query response before continuing.

Parameters

Name	Type	Description
name	String	Optional. The name of a field to query.
responseFunction	Function	The function called when the query results are available.
value	String	Optional. The field value to query for. Any pair of literals is considered a query pair (field : value).

Returns

Type	Description
void	

Example

The following is a basic example with a response function.

```
var rec = new GlideRecord('incident');
rec.query(recResponse);

function recResponse(rec) {
  while (rec.next()) {
   alert(rec.number + ' exists');
  }
}
```

The following example shows the difference between synchronus and asynchronous syntax, with an asynchronous example.

```
// synchronous call syntax (no response function): DO NOT
USE
        query();

        // asynchronous call syntax
        // performs query with current conditions, calls r
esponseFunction when done
        query(responseFunction)

        // synchronous call syntax (no response function)
: DO NOT USE
        // adds "category=hardware" to current query condi
tions and performs query
        query('category', 'hardware')

        // asynchronous call syntax
        // adds "category=hardware" to current query condi
tions, performs query, and calls responseFunction when don
e
        query('category', 'hardware', responseFunction)

        // asynchronous call example
        // adds "user_name=abel.tuter" to current query co
nditions, performs query, and calls defined response funct
ion when done
        function onLoad() {
        var now_GR = new GlideRecord("sys_user");
        now_GR.query("user_name", "abel.tuter", function(n
ow_GR) {
        if (now_GR.next()) {
        alert("You can access fields by name from the clie
nt API, just like in the server API:\n now_GR.name = " + n
ow_GR.name);
        alert("You can also access fields using getValue()
:\n now_GR.getValue(\"email\") = " + now_GR.getValue("emai
l"));
        if (now_GR.getDisplayValue) {
        alert("In Service Portal, Mobile, and Agent Worksp
ace, you can access a field's display value:\n now_GR.getD
isplayValue(\"company\") = " + now_GR.getDisplayValue("com
pany"));
        } else {
        alert("On the desktop, you cannot access a field'
s display value, but can get its sys_id:\n now_GR.company
= " + now_GR.company);
        }
        alert("You cannot dot-walk in the client API:\n no
w_GR.company.name = " + now_GR.company.name);
        }
        });
        }
```

Client side GlideRecord - setEncodedQuery(String encodedQuery)

Adds a specified encoded query string to the current query clause.

This method enables you to specify complex filters (encoded query strings) in a single query call, unlike other client-side addQuery() methods. Once you define all of the desired queries, call the Client side GlideRecord - query(String name, Function responseFunction, String value) to execute the specified query clause (filter). For additional information on encoded query strings, refer to Encoded query strings.

If you call this method multiple times before calling the query() method, the queries are AND'ed together.

Parameters

Name	Type	Description
encodedQuery	String	Encoded query string to add to the current query clause.

Returns

Type	Description
void	

Example

```
var now_GR = new GlideRecord('incident');
now_GR.setEncodedQuery('priority=4^ORstate=3');
now_GR.query(response);
```

```
function response(result) {
    while(result.next()) {
        console.log(result.getValue('number'));
    }
}
```

Client side GlideRecord - setLimit(Number maxQuery)

Sets the limit for how many records are in the GlideRecord.

Parameters

Name	Type	Description
maxQuery	Number	The limit for the number of records to retrieve.

Returns

Type	Description
void	

GlideURLV3 - Client

Provides methods for manipulating a URI.

The GlideURLV3 API can be used in client-side scripts using ListV2 and ListV3 APIs.

Note: This API is not supported by Service Portal, Now Mobile, or Agent Workspace.

GlideURLV3 - GlideURL(String contextPath)

Creates an instance of the GlideURL class.

Parameters

Name	Type	Description
contextPath	String	A relative path for the URL.

GlideURLV3 - addParam(String name, String value)

Adds a query string name-value pair to the URL.

Parameters

Name	Type	Description
name	String	Name of the query string parameter.
value	String	Query string value.

Returns

Type	Description
String	The GlideURL

Example

```
var gu = new GlideURL('incident.do');
var url = gu.addParam('sys_id', '-1');
```

GlideURLV3 - getURL(Object additionalParams)

Get the entire context path and query string parameters as a single URI.

Parameters

Name	Type	Description
additionalParams	Object	A name-value pair object that contains parameters that are

Name	Type	Description
		added to this URL request only. These additional parameters are not saved to the GlideURL object.

Returns

Type	Description
String	The GlideURL with the specified additional parameters added to the end.

GlideUser - Client

The GlideUser API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to get user information.

The GlideUser methods and properties are accessed through a global object (g_user) that is only available in client scripts. GlideUser

- contains name and role information about the current user.
- is typically used in client scripts and UI policies but is also found in UI actions that run on the client.
- cannot be used in business rules or UI actions that run on the server.
- avoids the need for GlideRecord queries to get user information.

Session information about the current user and current user roles is contained in the client (web browser). All GlideUser methods except getClientData() access the session information that is available by default. The getClientData() method requires setup on the server and use of putClientData() to make session information available.

For information on using client-side scripts, see Introduction to Client-side Scripting.

GlideUser - firstName

Returns the current user's first name.

Field

Name	Type	Description
firstName	String	Current user's first name.

Example

```
alert('first name = ' + g_user.firstName);
```

GlideUser - getClientData(String key)

Returns a client value set using setClientData() or GlideSession -- putClientData().

Session client data is a set of named strings that may be setup on the server using GlideSession -- putClientData(). You can use getClientData() during form load time to get information that the client script needs to make decisions about the form. For example, to identify which fields should be visible.

See also GlideForm.

Parameters

Name	Type	Description
key	String	Name of the client data to retrieve.

Returns

Type	Description
String	Value of the client data.

Example

```
var loginLanguage = g_user.getClientData("loginlanguage");
```

GlideUser - getFullName()

Returns the first and last name of the current user.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The current user's full name.

Example

```
var formalName = g_user.getFullName();
```

GlideUser - hasRole(String role, Boolean includeDefaults)

Returns true if the current user has the specified role or the admin role.

Parameters

Name	Type	Description
role	String	Role to check.
includeDefaults	Boolean	Optional. Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see

Name	Type	Description
		Roles. Default: false

Returns

Type	Description
Boolean	Returns true if the current user has the specified role or the admin role; otherwise returns false.

Example

```
var isInternal = g_user.hasRole('snc_internal', true);
```

Example

```
var isItil = g_user.hasRole('itil');
```

GlideUser - hasRoleExactly(String role, Boolean includeDefaults)

Determines whether the current user has the specified role.

Parameters

Name	Type	Description
includeDefaults	Boolean	Optional. Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see Explicit roles.

Name	Type	Description
		Default: false
role	String	Role to check.

Returns

Type	Description
Boolean	Returns true if the current user has the specified role.

Example

```
var isInternal = g_user.hasRoleExactly('snc_internal', tru
e);
```

Example

```
var isItil = g_user.hasRoleExactly('itil');
```

GlideUser - hasRoleFromList(String roles, Boolean includeDefaults)

Returns true if the current user has at least one of the specified roles or has the admin role.

Parameters

Name	Type	Description
roles	String	Comma-separated list of roles to check
includeDefaults	Boolean	Optional. Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see

Name	Type	Description
		Roles. Default: false

Returns

Type	Description
Boolean	Returns true if the current user has a role in the list or the admin role.

Example

```
var isOK = g_user.hasRoleFromList("itil, maint");
```

Example

```
var isOK = g_user.hasRoleFromList("itil, maint, snc_intern
al", true);
```

GlideUser - hasRoles(Boolean includeDefaults)

Returns true if the current user has any role.

Parameters

Name	Type	Description
includeDefaults	Boolean	Optional. Flag that indicates whether to include default roles, such as snc_internal and snc_external, in the request. For additional information on roles, see Roles.

Name	Type	Description
		Default: false

Returns

Type	Description
Boolean	Returns true if the current user has at least one role.

Example

```
var yesRole = g_user.hasRoles();
```

Example

```
var yesRole = g_user.hasRoles(true);
```

GlideUser - lastName

The current user's last name.

Field

Name	Type	Description
lastName	String	Current user's last name.

Example

```
alert('last name = ' + g_user.lastName);
```

GlideUser - setClientData(String key, String value)

Sets a client value that you can retrieve using getClientData().

See also GlideForm.

Parameters

Name	Type	Description
key	String	Name of the client data to store as a key.
value	Number	Value to assign to the key.

Returns

Type	Description
None	

Example

```
function onSubmit() {

        if (!g_user.getClientData('keyName')) {
                var now_GR = new GlideRecord('incident');
                now_GR.addActiveQuery();
                now_GR.setLimit(1);
                now_GR.query(cb);
                return false;
        }
        return true;
}

function cb(now_GR) {
        // <insert glide operation >
        g_user.setClientData('keyName', now_GR.getValue('<
number>'));
        g_form.submit();
}
```

GlideUser - userName

This property is the current user's username, for example gsmith02. It is not the user's name, for example George Smith.

Field

Name	Type	Description
userName	String	Current user's username.

Example

```
var userName = g_user.userName;
   alert('Current user = ' + userName);
```

GlideUser - userID

Returns the sys_id of the current user.

Field

Name	Type	Description
userID	String	sys_id of the current user.

Example

```
var userID = g_user.userID;
   alert('Current user ID = ' + userID);
```

GlideUIScripts - Client

Access UI scripts from within client-side code.

There is no constructor for this class. Access methods using the g_ui_scripts global object in any client-side code, such as client or validation scripts.

If calling a UI script with UI Type set to Mobile / Service Portal, use the `g_ui_scripts['nameOfScript'];` syntax. If calling a UI script with the UI Type set to All or Desktop, use the getUIScript() method to load the script. However, this method is not supported in Internet Explorer 11 when called outside of the Angular application environment. If calling a UI script outside of an Angular context using IE11, you must call the script directly.

Note: This class does not support UI scripts with the **Global** field set to true.

GlideUIScripts - getUIScript(String scriptName)

Calls a UI script with the UI Type set to All or Desktop from a client script or other client-side code. Returns a promise.

Use the `then()` function to perform an asynchronous action after the call resolves.

Note: This method is not supported in Internet Explorer 11 when called outside of the Angular application environment. If calling a UI script outside of an Angular context using IE11, call the script directly using the `g_ui_scripts['nameOfScript'];` syntax.

Parameters

Name	Type	Description
scriptName	String	API name of the UI script to run.

Returns

Type	Description
Promise	The result of the asynchronous call.

Example

```
function onLoad() {
    //Call the UI script directly If the UI Type is Mobil
e / Service Portal, for example:
    //g_ui_scripts['myUIScript'];

    //Use the method if the UI Type is All or Desktop
    g_ui_scripts.getUIScript('myUIScript').then(function(s
cript) {
        script.myUIScriptMethod();
    }, function() {
        console.log('The script did not load');
    });
}
```

Guided Tours - Client

Provides methods for launching and stopping guided tours.

This API includes methods used in Guided Tour Designer.

Guided Tours - applyListFilter(Function filter_func)

Sets a function to retrieve filtered tour results when the `getAllTours()` method is called.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
filter_func	Function	Filter function that takes a single `tour` object from the `tours[]` arra from `getAllTours()` method.

Returns

Type	Description
None	

Example

The following example shows basic API usage.

```
//create a filter function
var filtFunction = function(tour) {
   //only return those tours whose name starts with ‘my’
   return tour.name.indexOf(‘my’) === 0);
}
```

```
//apply the filter
top.NOW.guided_tours.api.applyListFilter (filtFunction);

//call the getAllTours method to observe the filtered tour
s
top.NOW.guided_tours.api.getAllTours (function(er, tours)
{
  if(!er) {
    console.log('The filtered tours are: ');
    console.log(tours);
  }
});
```

Example

The following example shows how to use the `options` field on the tour object to add JSON with custom tour identifiers for reading and filtering tours inside the `filter_func()` function.

```
top.NOW.guided_tours.api.applyListFilter(function(tour) {
       var options = (tour.options)? JSON.parse(tour.optio
ns): null;
       return (options && options.my_param) ? (options.my_
param == my_value) : false;
});
```

Guided Tours - endTour()

Stops a currently playing tour. This method silently exits if no tours are playing.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Null	

Example

```
//create a callback function to end the tour if it starts
correctly
var cbFunction = function(err) {
        if (err) {
   console.log('Error Occurred');
}
        else {
           // tour has started so we can call endTour
           top.NOW.guided_tours.api.endTour();
}
}

//calling the startTour method so that we can end the tour
 as soon as it starts
top.NOW.guided_tours.api.startTour('a297e04b732313007077ed
cc5ef6a780', 2, cbFunction);
```

Guided Tours - events.off(String event_name, Function listener_function)

Removes an existing event listener.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
event_name	String	Event name to be removed from the listener. Valid event names: • tourStarted

Name	Type	Description
		• tourEnded • tourCompleted • tourFailed • tourAbandoned • tourDismissed • stepStarted
listener_function	Function	Optional. If provided, specified listener function is removed from rem listeners attached with that event. If not provided, all listener functio that event are removed.

Returns

Type	Description
None	

Example

```
//create a callback function to handle the result of the a
pi call
var eventListenerTourStarted = function() {
   console.log('The tour has started');
}
var eventListenerTourEnded = function() {
   console.log('The tour has ended');
}

//attaching event listeners for tourStarted and tourEnded
Events
top.NOW.guided_tours.events.on('tourStarted',eventListener
TourStarted);
top.NOW.guided_tours.events.on('tourEnded', eventListenerT
ourEnded);


…
```

```
//start a tour
top.NOW.guided_tours.api.startTour ('a297e04b732313007077e
dcc5ef6a780', 2, cbFunction);
//As soon as the tour starts the eventListenerTourStarted
gets fired
…
top.NOW.guided_tours.api.endTour();
// eventListenerTourEnded gets fired

….

//removing the event listeners top.NOW.guided_tours.events
.off('tourStarted',eventListenerTourStarted);
top.NOW.guided_tours.events.off('tourEnded', eventListener
TourEnded);
```

Guided Tours - events.on(String event_name, Function listener_function)

Attaches an event listener to a guided tour event.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
event_name	String	Event name to be attached to the listener. Valid event names: • stepStarted • tourStarted • tourEnded • tourCompleted • tourFailed • tourAbandoned

Name	Type	Description
		• tourDismissed
listener_function	Function	Listener to be added. **Note:** Clear any event listener after it solves its purpose.

listener_function.obj	Object	Passed to `listener_function()` by each event in the following for
		• For stepStarted events:
		`{tour: '<tour_sys_id>', step: step_num}`
		• For all other events:
		`{tour: '<tour_sys_id>'}`
		JSON parameters:
		• tour_sys_id: String. Guided tour ID from the Guided Tours [sys_embedded_tour_guide] table
		• step_num: Number. Value between 0 (first step) and n (final step)

Example

The following example shows basic API usage.

```
//create a callback function to handle the result of the a
pi call
var eventListenerTourStarted = function() {
   console.log('The tour has started');
}
var eventListenerTourEnded = function() {
   console.log('The tour has ended');
}

//attaching event listeners for tourStarted and tourEnded
Events
top.NOW.guided_tours.events.on('tourStarted',eventListener
TourStarted);
```

```
top.NOW.guided_tours.events.on(‘tourEnded’, eventListenerT
ourEnded);

…
//start a tour
top.NOW.guided_tours.api.startTour ('a297e04b732313007077e
dcc5ef6a780', 2, cbFunction);
//As soon as the tour starts the eventListenerTourStarted
gets fired
…
top.NOW.guided_tours.api.endTour();
// eventListenerTourEnded gets fired

….

//removing the event listeners top.NOW.guided_tours.events
.off(‘tourStarted’,eventListenerTourStarted);
top.NOW.guided_tours.events.off(‘tourEnded’, eventListener
TourEnded);
```

Example

The following example shows how to use the `listener_function` parameter with `obj` as an argument.

```
top.NOW.guided_tours.events.on("tourStarted", function (ob
j){console.log(obj);});
```

Guided Tours - getAllTours(Function cb_function)

Gets a list of tours on the current page from which this method is called. Because this method is asynchronous, a callback function must be passed to determine operation success and get a list of tours.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
cb_function	Function	Callback function called by `getAllTours()` after attempt to fetch current page from which `getAllTours()` method is called.
cb_function.err	Object	Points to the error object if any occurred during the operation: `err = { success: false, message: 'string containing the object' }` Null otherwise.
cb_function.tours	Array	List of available tours for the page. If no tours are present on the page, `cb_function.tours` returns unc `if(!tours) console.log('No tour present')`

Returns

Type	Description
None	

Example

```
//create a callback function to handle the result of the A
PI call
var cbFunction = function(err, tours) {
        if (err) {
   console.log('Error Occurred');
}
        else {
            if(!tours) console.log('No tour present')
   else {
      tours.forEach(function(t) {
               console.log(t);
             });
          }
}
}
//calling the getTours method
top.NOW.guided_tours.api.getAllTours(cbFunction);
```

Guided Tours - loadPlayer()

Loads the guided tours player on a page in which guided tours player is not present by default.

Complete signature:

```
NOW.guided_tours.api.loadPlayer()
```

Parameters

Name	Type	Description
None		

Returns

Type	Description
None	

Guided Tours - startTour(String tour_id, Number step_number, Function cb_function)

Starts a tour. Because this method is asynchronous, you must pass a callback function to determine operation success.

Complete signature includes `top.NOW.guided_tours.api` preceding the method name.

Parameters

Name	Type	Description
tour_id	String	Sys ID of the tour from the Guided Tours [sys_embedded_tour_guide
step_number	Number	Optional. Step at which to start the tour. If not provided (or step num starts from the beginning.

Name	Type	Description
cb_function	Function	Optional. Callback function called by `startTour()` method after a launch the tour.
cb_function.err	Object	Points to the error object if any occurred during the operation: `err = { success: false, message: 'string containing the object' }` Null otherwise.

Returns

Type	Description
None	

Example

```
//create a callback function to handle the result of the A
PI call
var cbFunction = function(err) {
        if (err) {
   console.log('Error Occurred');
}
        else {
   console.log('The tour with tourid=%s was successfully l
aunched', tourId);
}
}

//calling the startTour method
top.NOW.guided_tours.api.startTour('a297e04b732313007077ed
cc5ef6a780', 2, cbFunction);
```

helpers - UI Builder

The helpers API provides general functionality that is common across page scripts, eliminating the need to write scripts for simple functionality such as opening and closing a modal.

This API is only available to page scripts, it is not available in any other UI Builder scripts including:

- component property value scripts
- component visibility scripts
- event payload scripts
- UX client script includes

helpers - helpers.modal.close(String modalId)

Closes the specified modal on the current page.

Parameters

Name	Type	Description
modalId	String	Modal component ID of the modal to close. Component IDs are auto generated when a component is dragged and dropped on the UI Builder stage. You can locate the ID on the property page.

Returns

Type	Description
None	

Example

This example shows closing a modal with a component ID that ends with `alert-modal`.

```
function handler({api, event, imports, helpers}) {
  helpers.modal.close("[component-id$='alert_modal']")
}
```

helpers - helpers.modal.open(String modalId, Object options)

Opens the specified modal on the current page.

You can only display one modal at a time within a page. If a modal is currently open, and you call this method, the existing modal is hidden and the new modal appears.

Parameters

Name	Type	Description
modalId	String	Component ID of the modal to open. Component IDs are auto generated when a component is dragged and dropped on the UI Builder stage. You can locate the ID on the property page.

options	Object	Optional.
		`"options": {`
		`  "content": "String",`
		`  "contentFullWidth": Boolean,`
		`  "headerLabel": "String",`
		`  "size": "String"`
		`}`

options.content	String	Text content for the modal.
options.contentFullWidth	Boolean	Flag that indicates whether to remove the horizontal padding around the body of the modal in order to fit wider content. Valid values: • true: Remove the padding. • false: Do not remove the padding. Default: false
options.headerLabel	String	Text content for the modal header.

Name	Type	Description
options.size	String	Size of the modal container. Most sizes automatically expand to fill the viewport when the screen size is small. Valid values: • sm • md • lg • fullscreen Default: sm

Returns

Type	Description
None	

Example

This example shows opening a modal with ca omponent ID that ends with `alert-modal`.

```
function handler({api, event, imports, helpers}) {
  helpers.modal.open("[component-id$='alert_modal']")
}
```

helpers - helpers.navigate.setRouteParams(Object params)

Passes the specified parameters down to other components on the same page.

Use this method in any page component that wants to add a parameter in a URL. You may want to add a parameter to a URL when another component needs to know the current value of that parameter when it changes, so it can react to it. For example, use this method to pass the `selectedIndex` of a tab component so it reflected in the URL to give focus to that tab.

Parameters

Name	Type	Description
params	Object	Key-value pairs of optional parameters to pass to other components. This must be a plain, flat object with only primitive values. Array or object references are ignored and not added to the URL. All specified keys must be part of the optional parameters in the route configuration or they are also ignored. For additional information on optional parameters, see Create a page in UI Builder.

Returns

Type	Description
None	

Example

This code example shows how to append the URL `params/selected-tab-index/2`. Note that the parameter in the actual URL is changed from camel case to snake case, so `selectedTabIndex` becomes `selected-tab-index`.

```
function handler({api, event, helpers, imports}) {
  helpers.navigate.setRouteParams({'params': {'selectedTab
Index':  2}});
}
```

helpers - helpers.navigate.to(String route, Object fields, Object params, Boolean redirect, Boolean passiveNavigation, String targetRoute)

Navigates from one screen to another based on the specified route and field information. URL changes and the respective screen loads are observed.

Parameters

Name	Type	Description
route	String	Name of the route. Must be a valid entry from UX App Routes (sys_ux_app_route.list). This value is reflected in the URL, and the URL is created based on the route, fields, and optionalParams column values: `/<route>/<field1Value>/{<field2Value>/params/<optionalParamKey1>/<optionalParamValue1>/<optionalParamKey2>/<optionalParamValue2>` For example: `/record/incident/12345/params/selectedTabIndex/4`
fields	Object	Optional. Key-value pairs of required parameters. For example: `'fields' : {'table': 'incident', 'sysId': '12345'}`.
params	Object	Optional. Key-value pairs of optional parameters. For example: `'params' : {'selectedTabIndex': 4}`.
redirect	Boolean	Flag that indicates whether to remove the latest history entry from the browser history. For example, if you navigate to sites A, B and then C. If redirect is set to `true` while navigating to C, the history entry for B is removed. The browser history only shows only A and C.

Name	Type	Description
		Valid values: • true: Removes the latest history entry, and redirects to the latest URL. • false: Does not remove any history entries. Default: false
passiveNavigation	Boolean	Flag that indicates whether to perform background navigation. Background navigation is when a page is opened but it is not active or shown. For example, opening an inactive tab for the page, but it is not visible but loaded in the background. Valid values: • true: Perform background navigation. • false: Do not perform background navigation. Default: false
targetRoute	String or Object	Sub navigation to a drill-down, deep-link, or sub-tab. If set to `current`, the current route does a sub-navigation under the current URL. For example, if `/record/incident/123` is the current URL, and the following call is made: `helper.navigate.to('record', {'table': 'problem', 'sysId': '567'}, {}, false, false, 'current');` The following URL is generated: `/record/incident/123/sub/record/problem/567`

Name	Type	Description
		Note: targetRoute can be either a string such as `'current'` or an object, such as `navigation NAV_ITEM_SELECTED payload`.

Returns

Type	Description
None	

Example

This example shows how to navigate to a page passing just the route parameter.

```
function handler({api, event, imports, helpers}) {
  helpers.navigate.to('test');
}
```

Example

This example shows how to navigate to a page passing the route and fields parameters.

```
function handler({api, event, imports, helpers}) {
  helpers.navigate.to('test', {'key': 'value'});
}
```

Example

This example shows how to navigate to a page passing the route, fields, and params parameters.

```
function handler({api, event, helpers, imports}) {
  helpers.navigate.to('test', {'key': 'value'}, {'first':
'FirstName', 'last': 'Last Name'});
}
```

helpers - helpers.screen.updateStatus(Object statusObj)

Enables pages to report their status updates, such as title, icon, dirty state, message, and error changes.

Status updates are reported to WorkspaceChrome or AppShell, whichever the outer layer is, and acting as the host.

Parameters

Name	Type	Description
statusObj	Object	Payload to send to the current page to report that the content has been updated. Valid values: • dirtyModalId: (String) ID of the modal that has changed. • hasError: (Boolean) Flag that indicates that there are errors on the page. • hasUpdate: (Boolean) Flag that indicates that there were updates to the page. • icon: (String) Name of the updated or new icon. • isDirty: (Boolean) Flag that indicates whether the page is dirty (values have changed). • message: (String) Updated/new message. • screenKey: (String) ID of the screen on which the change occurred. Every screen has a screenKey as a property on the screen macroponent inside sn-canvas-screen. • status: (String) Status operation for this action. This value must be one of the following: inserted, deleted, saved, or closed. • title: (String) Updated/new display title.

Name	Type	Description
		• tooltipPreview: (JSON) Updated or new tool tip. For example, `tooltipPreview : { primaryTitle, secondaryContent: {} }`

Returns

Type	Description
None	

Example

```
screen.updateStatus({'dirtyModalId': 'customModalId', 'isD
irty': true});
```

helpers - helpers.snHttp(String url, Object options)

Makes an HTTP request to the ServiceNow instance and returns a promise with the results.

Note: There is a known issue where objects named options are omitted from the HTTP response.

```
{
  options: {},
  otherFields: {}
}

becomes

{
  otherFields: {}
}
```

Parameters

Name	Type	Description
url	String	HTTP endpoint relative to the instance URL. For example, `/api/now/table/incident` or `/api/now/table/incident/a83820b58f723300e7e16c7827bdeed`.

options	Object	Describes the contents of the HTTP request.
		`"options": {`
		`  "batch": Boolean,`
		`  "body": {Object},`
		`  "headers": {Object},`
		`  "method": "String"`
		`}`

options.batch	Boolean	Optional. Flag that indicates whether this HTTP request should be batched with other HTTP requests made to the instance. Valid values: • true: Make request as part of a batch request. • false: Make dedicated request. Default: true
options.body	Object	Optional. Data to send as the request body. Only applicable for request methods `PUT`, `POST`, and `PATCH`. Elements in the object depend on the type of HTTP method. For details, refer to the code examples below. Default: `{}`
options.headers	Object	Optional. Additional HTTP request headers.

Name	Type	Description

		For example:
		`headers: {`
		`  "Content-Type": "application/json",`
		`  "Accept": "application/xml"`
		`}`

options.method	String	Optional. HTTP method. Valid values: • DELETE • GET • PATCH • POST • PUT Default: GET

Returns

Type	Description

error	Object that describes any error returned by the REST API.
	
	Data type: Object
	`"error": {`
	`  "data": "String",`
	`  "message": "String",`
	`  "options": {Object},`
	`  "status": Number,`
	`  "statusText": "String"`
	`}`

error.data	Response returned from the HTTP API.

Type	Description
	Data type: Defined by REST API
error.message	Message describing the error encountered when trying to process the HTTP request. **Note:** This parameter is not always returned. Data type: String

error.options	Describes the original HTTP request.
	Data type: Object
	`"options": {`
	`  "headers": {Object},`
	`  "responseHeader": {Object}`
	`}`

error.options.headers	Object containing a list of all of the request headers sent in the request. Data type: Object
error.options.responseHeaders	Object containing a list of all of the response headers sent in the request. Data type: Object
error.status	Returned HTTP error status code, such as 400, 405, or 500. Data type: Number
error.statusText	Returned HTTP status message, such as Bad Request.

Type	Description
	Data type: String
response	Returned when HTTP request is successful. The response to the HTTP request. Data type: Any

Example

The following example show how to call snHttp() which returns a promise.

```
function handler({api, event, helpers, imports}) {
  helpers.snHttp('/api/now/table/u_movie', {method: 'GET'}
)
    .then(({response}) => {
      // do something with the table data
    })
    .catch(({error}) => {
      const message = `Error: ${error.data.error.message}`
;
      console.error(message);
      api.emit('NOW_UXF_PAGE#ADD_NOTIFICATIONS', {
        id: 'alert5',
        status: 'high',
        icon: 'info-circle-outline',
        content: message,
        action: { type: 'dismiss' }
    });
  });
}
```

Example

The following example show how to call snHttp() using async and await.

```
async function handler({helpers}) {
  try {
    const result = await helpers.snHttp('/api/now/table/u_
movie', {method: 'GET'});
```

```
  } catch ({error}) {
      const message = `Error: ${error.data.error.message}`
;
      console.error(message);
      api.emit('NOW_UXF_PAGE#ADD_NOTIFICATIONS', {
        id: 'alert5',
        status: 'high',
        icon: 'info-circle-outline',
        content: message,
        action: { type: 'dismiss' }
      });
  }
}
```

Example

The following example show how to set up a POST request.

```
function handler({api, helpers, event, imports}) {
  helpers
    .snHttp("/api/now/table/incident", {
      method: "POST",
      body: {
        description: "Sample description",
        close_notes: "Sample close notes",
        order: "-1"
      }
    })
    .then(({ response }) => {
      // handle POST request response
    })
    .catch(({ error }) => {
      // handle POST request errors
    });
}
```

Example

The following example show how to set up a PUT request.

```
function handler({api, helpers, event, imports}) {
  helpers
    .snHttp("/api/now/table/incident/a83820b58f723300e7e16
c7827bdeed2", {
      method: "PUT",
      body: {
        activity_due: "1970-04-02 18:26:17"
      },
      headers: {
        "Content-Type": "application/json",
        "Accept": "application/xml"
      }
    })
    .then(({ response }) => {
      // handle PUT request response
    })
    .catch(({ error }) => {
      // handle PUT request errors
    });
}
```

helpers - helpers.timing.clearInterval(Number timeoutId)

Cancels the execution of the function that was scheduled through a prior setInterval() call.

Parameters

Name	Type	Description
timeoutId	Number	Unique identifier of the scheduled function to clear. This value is returned by the corresponding setInterval() call.

Returns

Type	Description
None	

Example

This example shows using clearInterval() to cancel a timing operation that was previously set using thesetInterval() method. This function could be invoked by a user clicking a **Disable Auto-refresh** button on a page.

```
function handler({api, helpers}) {
  api.setState('intervalId', ({currentValue}) => {
    if (currentValue > -1) {
      helpers.timing.clearInterval(currentValue);
    }
    return -1;
  });
}
```

helpers - helpers.timing.clearTimeout(Number timeoutId)

Cancels the execution of the function that was scheduled through a prior setTimeout() call.

Parameters

Name	Type	Description
timeoutId	Number	Unique identifier of the scheduled function to clear. This value is returned by the corresponding setTimeout() call.

Returns

Type	Description
None	

Example

This code example shows how to terminate a timer with the specified timeoutId.

```
function handler({api, helpers}) {
  api.setState('timeoutId', ({currentValue}) => {
    if (currentValue > -1) {
      helpers.timing.clearTimeout(currentValue);
    }
    return -1;
  });
}
```

helpers - helpers.timing.setInterval(Function fn, Number delay)

Repeatedly executes the specified function, using the specified delay value as the interval between function calls.

Unlike the native JavaScript setInterval() method, this method does not support passing a code string to evaluate as the first argument. Any additional arguments are passed to the function itself.

Parameters

Name	Type	Description
fn	Function	Function to repeatedly execute.
delay	Number	Length of the time-interval between each function execution. Unit: Milliseconds

Returns

Type	Description
Number	Unique identifier of the function execution operation. Use this value in the helpers - helpers.timing.clearInterval(Number timeoutId) method if you need to cancel this operation.

Example

This code example shows how to refresh the timestamp on a page every second. This function could be invoked by a user clicking an **Enable Auto-refresh** button on a page.

```
function handler({api, helpers}) {
  // Every one second, refresh the value of current timest
amp client state parameter
  const intervalId = helpers.timing.setInterval(() => {
    api.setState('currentTimestamp', new Date().toString()
)
  }, 1000);
  // The interval ID is kept in state to use when calling
the helpers.timing.clearInterval() method at a later point
  api.setState('intervalId', intervalId);
}
```

helpers - helpers.timing.setTimeout(Function fn, Number delay)

Executes the specified function, after the specified delay.

Unlike the native JavaScript setTimeout() method, this method does not support passing a code string to evaluate as the first argument. Any additional arguments are passed to the function itself.

Parameters

Name	Type	Description
fn	Function	Function to execute.
delay	Number	Length of the time to wait before calling the specified function. Unit: Milliseconds

Returns

Type	Description
Number	Unique identifier of the function execution operation. Use this value in the helpers - helpers.timing.clearTimeout(Number timeoutId) method if you need to cancel this operation.

Example

This code example shows how to set a 20 minute timer. You could associate this function with a button **Remind me in 20 minutes**.

```
function handler({api, helpers}) {
  const timeoutId = helpers.timing.setTimeout(() => {
    api.emit('NOW_UXF_PAGE#ADD_NOTIFICATIONS', {
      id: 'alert5',
      status: 'high',
      icon: 'info-circle-outline',
      content: 'Try to look away at something that is 20 f
eet away from you for a total of 20 minutes.',
      action: { type: 'dismiss' }
    });
  }, 20 * 60 * 1000);

  // The timeout ID is kept in state to use when calling t
he helpers.timing.clearTimeout() method at a later point
  api.setState('timeoutId', timeoutId);
}
```

helpers - helpers.translate(String message, String tokens)

Asynchronously retrieves and translates the specified message based on the current user's session language.

You can use this method with the api - setState(String stateParam, Any value) to bind the translated value to other fields on the page.

Note: You can call this method using a promise or `async` and `await`. The code examples below show both implementations.

Parameters

Name	Type	Description
message	String	Message to translate.
tokens	String	Optional. Comma-separated list of parameters to use for replacing string variables. For example: `helpers.translate('Text {0} {1}', 'to', 'translate');`

Returns

Type	Description
String	Translated text string.

Example

The following example shows how to pass in table field references to embed in the corresponding variables in a string, using a promise.

```
function handler ({api, helpers}) {
  helpers.translate(‘Welcome {0} {1}!’, user.firstName, us
er.lastName)
    .then((translatedText) => {
      api.setState('greeting', translatedText);
    });
}
```

Example

The following example shows how to use `async` and `await` in your function instead of a promise.

```
async function handler ({api, helpers}) {
  const translatedText = await helpers.translate('Welcome
to {0}', 'ServiceNow');
    api.setState('greeting', translatedText);
}
```

i18N - Client

Provides methods to get and format translated messages.

The i18N methods are accessed using the g_i18n global object.

i18N - format(String message, Object map)

Formats a string containing named tokens with values from a map.

Parameters

Name	Type	Description
message	String	The message to have the tokens added.
map	Object	The map of name/ value pairs to replace in the message.

Returns

Type	Description
String	The formatted string

Example

```
// Returns: "The rich young ruler was very very rich"
nowapi.g_i18n.format("The {p1} {p2} {p3} was very very {p1
}",{p1: "rich", p2: "young", p3: "ruler"});
```

i18N - getMessage(String msgKey, Function callback)

Retrieves a translated message.

If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

Parameters

Name	Type	Description
msgKey	String	The message to be retrieved.
callback	Function	The function to be called when the

Name	Type	Description
		message has been retrieved. The callback function has one argument, a string that is the translated message.

Returns

Type	Description
void	

i18N - getMessages(Array msgKeys, Function callback)

Retrieves a set of messages.

If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

Parameters

Name	Type	Description
msgKeys	Array	An array of keys specifying the messages to be retrieved.
callback	Function	The function to be called when the messages have been retrieved. The callback function has one argument, an object containing key-value pairs, where key is the requested message

Name	Type	Description
		key, and the value is the translated string.

Returns

Type	Description
void	

openFrameAPI - Client

OpenFrame is an omni-present frame that communication partners can use to integrate their systems into the ServiceNow platform.

One of the core requirements is the ability to connect and serve code from different domains that can connect seamlessly with partner subsystems. This cross domain connection is required to keep connections and callbacks registered into communication systems without any cross domain issues.

OpenFrame has two significant parts: one that lives in the ServiceNow application (referred to as TopFrame) and this API that is sourced from the partner application. This API has the necessary methods to communicate with TopFrame and control the visual features of the OpenFrame.

> **Note:** To stay current with reference to the OpenFrame library, use the following resource URI: https://[servicenow instance]/scripts/openframe/latest/openFrameAPI.min.js.

APIs not supported on Agent Workspace

The following functionalities are not supported on Agent Workspace:

- openServiceNowList
- OpenCustomURL

openFrameAPI - hide()

Hides the OpenFrame in the TopFrame.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
openFrameAPI.hide()
```

openFrameAPI - init(Object config, function successCallback, function failureCallback)

Initialize OpenFrame, must be the first method called.

This method initializes communication to TopFrame and initializes any visual elements passed in the config parameter.

Parameters

Name	Type	Description
config	Object	An object of key value pairs. The possible keys are height, width, title, subTitle, and titleIcon. All keys are optional.
successCallback	function	The callback function used if the init method succeeds. The openframe configuration stored in the system is passed as a parameter to the callback function.

Name	Type	Description
failureCallback	function	The callback function used if the init method fails.

Returns

Type	Description
void	

Example

```
var config = {
height: 300,
width: 200
}
function handleCommunicationEvent(context) {
console.log("Communication from Topframe", context);
}
function initSuccess(snConfig) {
console.log("openframe configuration",snConfig);
//register for communication event from TopFrame
openFrameAPI.subscribe(openFrameAPI.EVENTS.COMMUNICATION_E
VENT,
handleCommunicationEvent);
}
function initFailure(error) {
console.log("OpenFrame init failed..", error);
}
openFrameAPI.init(config, initSuccess, initFailure);
```

openFrameAPI - isVisible(function callback)

Checks to see if the OpenFrame is visible in the TopFrame.

Parameters

Name	Type	Description
callback	function	The callback function receives a parameter with a value of true or false. True if OpenFrame is visible and false if not visible.

Returns

Type	Description
void	

Example

```
function callback(isVisible) {
console.log(isVisible)
}
openFrameAPI.isVisible(callback)
```

openFrameAPI - openCustomURL(String details)

Opens a custom URL in TopFrame.

Note: This API is not supported on Agent Workspace.

Parameters

Name	Type	Description
Url	String	A string of 2083 or fewer characters.

Returns

Type	Description
void	

Example

```
openFrameAPI.openCustomURL('10_cool_things.do');
```

openFrameAPI - openServiceNowForm(Object details)

Opens a form URL.

When an agent receives an incoming call, the OpenFrame window displays information such as the account, contact, or consumer. Clicking a link in the OpenFrame window displays the corresponding record.

- In the platform interface, this API opens a form URL in TopFrame.
- For Agent Workspace, this API supports interaction tab management. In Agent Workspace, an interaction record opens in a parent tab and the specified entity record opens in a child tab under the interaction tab.

Parameters

Name	Type	Description

details	Object	Key-value pairs that identify the form URL to open.
		`"details": {`
		`  "entity": "String";`
		`  "interaction_sys_id": "String";`
		`  "query": "String"`
		`}`

details.entity	String	Table or entity name.

Name	Type	Description
details.interaction_sys_id	String	Optional. Sys_id of the interaction record to open as parent tab in Agent Workspace. **Note:** In the platform interface the interaction_sys_id is ignored.
details.query	String	Query to identify the record to open, such as: `query:'sys_id=<record_sys_id>'`.

Returns

Type	Description
void	

Example

The following example shows basic usage in platform:

```
openFrameAPI.openServiceNowForm({entity:'customer_account'
,
query:'sys_id=447832786f0331003b3c498f5d3ee452', 'interact
ion_sys_id':'3be092313b711300758ce9b534efc4dd'});
```

Example

The following example shows how to use the query parameter to create a new record with data provided in the form by using sysparm_query and an encoded query to populate the first and last name fields in Workspace:

```
openFrameAPI.openServiceNowForm({ entity: 'sys_user',
query: 'sys_id=-1&sysparm_query=first_name=Ivan^last_name=
Greggor' });
```

openFrameAPI - openServiceNowList(Object details)

Opens a list URL in TopFrame.

Note: This API is not supported on Agent Workspace.

Parameters

Name	Type	Description
details	Object	An object of key value pairs. The possible keys are • entity, the table name • query, an encoded query string

Returns

Type	Description
void	

Example

```
openFrameAPI.openServiceNowList({entity:'case', query:'act
ive=true'});
```

openFrameAPI - setFrameMode(mode)

Sets the OpenFrame mode.

The mode passed in this API:

- Sets the appropriate icon in the header: collapse or expand

- Raises the relevant event for CTI:
  - openFrameAPI.EVENTS.COLLAPSE
  - openFrameAPI.EVENTS.EXPAND

Parameters

Name	Type	Description
Mode	String	Set OpenFrame Mode. Enumerated options: 1. openFrameAPI.FRAME_MODE.COLLAPSE 2. openFrameAPI.FRAME_MODE.EXPAND

Returns

Type	Description
void	

Example

```
openFrameAPI.setFrameMode(openFrameAPI.FRAME_MODE.COLLAPSE
);
```

openFrameAPI - setHeight(height)

Sets the OpenFrame height.

Parameters

Name	Type	Description
Height	Number	Height in pixels

Returns

Type	Description
void	

Example

```
openFrameAPI.setHeight(100);
```

openFrameAPI - setIcons(Array icons)

The OpenFrame header can include icons that are placed next to the close icon.

Parameters

Name	Type	Description
icons	Array	A list of icon configurations, where each icon configuration is an object with key values imageURL, imageTitle, and any other needed context.

Returns

Type	Description
void	

Example

```
openFrameAPI.setIcons([{imageURL:'https://mydomian.com/ima
ge/mute.png',
imageTitle:'mute', id:101}, {imageURL:'https://mydomian.co
m/image/hold.png',
imageTitle:'hold', id:102}]);
```

openFrameAPI - setPresenceIndicator(presence)

Sets the presence indicator to display agent availability in a workspace.

For more information on configuring OpenFrame, refer to Create an OpenFrame configuration

Parameters

Name	Type	Description
state	String	Presence state of the agent. Default states: • Available • Away • Offline You can also specify custom states.
color	String	Presence indicator color on workspace. Supported colors: • red • orange • grey

Name	Type	Description
		• green

Returns

Type	Description
void	

Example

```
openframeAPI.setPresenceIndicator('Available', 'green');
```

openFrameAPI - setSize(Number width, Number height)

Sets the OpenFrame size.

Parameters

Name	Type	Description
width	Number	Should be greater than zero.
height	Number	Should be greater than zero.

Returns

Type	Description
void	

Example

```
openFrameAPI.setSize(300, 370);
```

openFrameAPI - setSubtitle(String subTitle)

Sets the OpenFrame subtitle.

Parameters

Name	Type	Description
subTitle	String	A string of 256 or fewer characters.

Returns

Type	Description
void	

Example

```
openFrameAPI.setSubtitle('+18888888888');
```

openFrameAPI - setTitle(String title)

Sets the OpenFrame title.

Parameters

Name	Type	Description
title	String	A string of 256 or fewer characters.

Returns

Type	Description
void	

Example

```
openFrameAPI.setTitle('Incoming Call');
```

openFrameAPI - setTitleIcon(Object icon)

Sets the OpenFrame's title icon.

Parameters

Name	Type	Description
icon	Object	Object of key value pairs. Keys include imageURL, imageTitle, and any other context needed.

Returns

Type	Description
void	

Example

```
openFrameAPI.setTitleIcon({imageURL:'/my/image/path.png',
imageTitle:'mute', id:101});
```

Example

```
openFrameAPI.setTitleIcon({imageURL:'https://mydomian.com/
image/path.png',
imageTitle:'mute', id:101});
```

openFrameAPI - setWidth(width)

Sets the OpenFrame width.

Parameters

Name	Type	Description
Width	Number	Width in pixels

Returns

Type	Description
void	

Example

```
openFrameAPI.setWidth(100);
```

openFrameAPI - show()

Makes the OpenFrame visible in the TopFrame.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

```
openFrameAPI.show()
```

openFrameAPI - subscribe(openFrameAPIEVENT event, function eventCallback)

Subscribes to the event.

Parameters

Name	Type	Description
event	openFrameAPIEVENT	One of the following events: • openframe_agent_off_interaction: Indicates the presence of an agent on chat as off or available. • openframe_awa_agent_presenc

Name	Type	Description
		e: In Advanced Work Assignment (AWA), this event occurs when there is any change in the agent presence state. Computer Telephony Integration (CTI) developers can subscribe to the this event to receive presence state changes. • openframe_awa_workitem_accepted: Occurs when a work item is accepted by an agent. • openframe_awa_workitem_offered: Occurs when a work item is offered to an agent. • openframe_awa_workitem_rejected: Occurs when a work item is rejected by an agent. • openframe_before_destroy: Occurs before the TopFrame is unloaded. • openframe_collapse: Occurs when the collapse icon is clicked on the OpenFrame header. • openframe_communication: Application-specific and can be customized. • openframe_communication_failure: Occurs when communication to TopFrame fails.

Name	Type	Description
		• openframe_expand: Occurs when the expand icon is clicked on the OpenFrame header. • openframe_header_icon_clicked: Deprecated. Use openframe_icon_clicked or openframe_title_icon_clicked instead. • openframe_hidden: Occurs when the OpenFrame is hidden. • openframe_icon_clicked: Occurs when any icon other than the close icon is clicked on the OpenFrame footer. The callback receives the icon object as a parameter. • openframe_shown: Occurs when the OpenFrame is shown. • openframe_title_icon_clicked: Occurs when the title icon is clicked on the OpenFrame. The callback receives the titleIcon object as a parameter.
eventCallback	function	Function called when the specified event occurs.

Returns

Type	Description
None unless described otherwise.	Most event subscriptions have no return values, with the following exception(s): In AWA, the openframe_awa_agent_presence event returns the presence object:

Type	Description
	• presence: Information about an agent's current presence state and channel. Output example below. • presence.name: Name of the agent's presence state. • presence.sys_id: Presence state sys_id. Located in the Presence States [awa_presence_state] table. • presence.available: Flag that indicates whether the agent is available. • presence.channels: List of objects that describe the available channels of communication with the agent. • presence.channels.name: Channel name, such as Chat or Phone. • presence.channels.available: Flag that indicates whether the channel is available. • presence.channels.sys_id: Channel sys_id. Located in the Service Channels [awa_service_channel] table. • presence.channels.restrict_update: Flag that indicates whether the user can restrict updates.

Example

Example

```
function handleIconClick(context) {
console.log("Icon was clicked", context);
}
openFrameAPI.subscribe(openFrameAPI.events.openframe_awa_a
gent_presence, handleIconClick);
```

Output

```
// Sample presence object output
// openframe_awa_agent_presence event only

{
   "result":{
      "presence":{
         "name":"Available",
         "sys_id":"<SysID>",
         "available":true,
         "channels":[
            {
               "name":"Chat",
               "available":true,
               "sys_id":"<SysID>",
               "restrict_update":false
            },
            {
               "name":"Phone",
               "available":true,
               "sys_id":"<SysID>",
               "restrict_update":false
            }
         ]
      }
   }
}
```

openFrameAPI - version()

Returns the OpenFrame API version.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The OpenFrame API version

Example

```
var version = openFrameAPI.version();

console.log("API version " + version);
```

NotifyClient - Client

The NotifyClient API allows you use Notify telephony functionality, such as making and receiving calls, from a web browser.

Several NotifyClient methods take a callback function as a parameter. Because NotifyClient calls are made asynchronously, these methods cannot return a value directly. Use the callback function to parse the returned data, such as by assigning variables or making other API calls.

NotifyClient - Client(Object notifyConfig, Boolean initializeVendorClientLazily)

Instantiates a new Notify WebRTC Client object.

Parameters

Name	Type	Description
initializeVendorClientLazily	Boolean	Flag that indicates whether to use the autoSelectVendorCallback function passed in the setCallerId() method to automatically set the caller's associated vendor (notifyConfig.vendor does not need to be defined in the constructor). • false: Default. Do not use the autoSelectVendorCallback function to

Name	Type	Description
		set the caller's vendor. The vendor must be set in the constructor. • true: Use the autoSelectVendorCallback function to define the vendor when the caller ID is set.
notifyConfig	Object	JSON object that contains the configuration settings for the Notify WebRTC Client.
notifyConfig.autoLoadScriptResources	Boolean	Flag that indicates how to load the core JS library needed by the vendor client. • false: Default. Use vendor specific codes to load the required vendor JS library (enables backwards compatibility). • true: Use notifyClient.js to load the core JS library.
notifyConfig.callerId	Number	Registered Notify number to use. Do not directly set this value. Use the method

Name	Type	Description
		notifyClient.setCallerID() to set this value.
notifyConfig.forceRefreshToken	Boolean	Flag that indicates whether to auto-renew expired client tokens. • false: Do not automatically renew client tokens when they expire. • true: Default. Automatically renew client tokens when they expire.
notifyConfig.skipParentId	Boolean	Flag that indicates whether to immediately invoke the onIncoming caller for incoming calls. • false: Default. Do not immediately invoke the onIncoming event handler. • true: Immediately invoke the onIncoming event handler. By setting this flag, if there is another call, where the `<Dial><Client>` Twiml caused the incoming call, then setting this flag causes the system

Name	Type	Description
		to auto poll the backend. This auto poll obtains the parent notify_call reference.
notifyConfig.vendor	Constant	Vendor to which the caller belongs. • SNC.Notify.Vendor.TWILIO_DIRECT • SNC.Notify.Vendor.TWILIO (older, deprecated Twilio driver)

Example

The following example shows how to create the NotifyClient constructor, register various event listeners, and initialize the client driver.

```
jQuery(function () {

  var notifyConfig = {
    autoLoadScriptResources: true // This will take care o
f auto loading the JS resources needed by the client (if a
ny)
  };
  var client = new SNC.Notify.Client(notifyConfig, true);
// The second argument ensures that the proper vendor for
the given number is auto determined
  client.setCallerId('valid_notify_long_number', function
() {
    // This is called after the vendor has been determined
.
```

```
    if (!notifyConfig.vendor) // Means this number has no 
compatible vendor
      return;

    client.addEventListener(SNC.Notify.STD_EVENTS.ONLINE, 
function () {
      // Ability to call is available
    });
    client.addEventListener(SNC.Notify.STD_EVENTS.OFFLINE
, function () {
      // Ability to call is _not_ available right now
    });
    client.addEventListener(SNC.Notify.STD_EVENTS.ERROR, f
unction (msg, code) {
      // Some error happened
    });
      //... register other event handlers here
      //Show UI elements which can be used to invoke clien
t.call() and other APIs
    client.init(); // This is important to call this.
    });
});
```

NotifyClient - addEventListener(String event, Function fn)

Registers an event handler to listen for changes in a Notify client.

Using this method you can register multiple listeners. Each listener must be a separate method call.

Parameters

Name	Type	Description
event	String	Name of the event to listen for. Instead of passing strings, use the constants defined in `SNC.Notify.STD_EVENTS`.

Name	Type	Description
		• CALL_START: call has started and is in progress. • CALL_CANCEL: caller canceled the call. • CALL_INIT: WebRTC connected to a call (incoming or outgoing). • CALL_DISCONNECT: current call has been disconnected. • ERROR: Error occurred. Parameters: `message(string)`, `errCode(string)` ◦ `message`: error message to display. ◦ `errCode`: Optional. Associated error code. • INCOMING_CALL: Call is coming in. Parameters: `from(string)`, `to(string)`, `callId(string)`, `parentId(string)`, `sysId(string)`, `isFromClient(boolean)`

Name	Type	Description
		<ul><li><ul><li>`from`: caller's phone number.</li><li>`to`: called phone number.</li><li>`callId`: SID of the call.</li><li>`parentId`: parent notify_call reference. If skipParentId is set to true, this parameter should not be passed.</li><li>`sysId`: WebRTC-to-WebRTC calls only. Unique identifier (sys_id) of the caller.</li><li>`isFromClient`: WebRTC-to-WebRTC calls only. Flag that indicates whether the call is from another WebRTC client.</li></ul></li><li>CALL_MUTE: client is muted.</li><li>CALL_UNMUTE: client is unmuted.</li><li>OFFLINE: WebRTC session is not active.</li><li>ONLINE: WebRTC session is ready. Must</li></ul>

Name	Type	Description
		be set after calling the init() method.

Returns

Type	Description
Function	Function to use to de-register a listener.

Example

This example shows how to register multiple listeners.

```
jQuery(function () {

  var notifyConfig = {
    autoLoadScriptResources: true // This will take care o
f auto loading the JS resources needed by the client (if a
ny)
  };
  var client = new SNC.Notify.Client(notifyConfig, true);
// The second argument ensures that the proper client for
the given number is auto determined
  client.setCallerId('valid_notify_long_number', function
() {
    // This is called after the client has been determined
.

    if (!notifyConfig.vendor) // Means this number has no
compatible client
      return;

    client.addEventListener(SNC.Notify.STD_EVENTS.ONLINE,
function () {
      // Ability to call is available
    });
    client.addEventListener(SNC.Notify.STD_EVENTS.OFFLINE
, function () {
      // Ability to call is _not_ available right now
    });
    client.addEventListener(SNC.Notify.STD_EVENTS.ERROR, f
unction (msg, code) {
      // Some error happened
    });
      //... register other event handlers here

    client.init(); // This is important to call this.
    });
});
```

Example

This example shows how to de-register a listener.

```
var dereg = notifyClient.addEventListener(SNC.Notify.STD_E
VENTS.ONLINE, function () {
 ...
 });
 dereg();
  // The event listener function is no longer triggered.
```

NotifyClient - call(Object identifier)

Calls the specified phone number or the phone number associated with a specified user.

Note: When checking the status of a call/connection, always compare against the constants provided by `SNC.Notify.Status`.

Parameters

Name	Type	Description
identifier	Object	JSON object that contains either a phone number to call or the sys_id of a WebRTC user. Passing a user sys_id causes the call to be made through

Name	Type	Description
		browser-to-browser communication. You can obtain the user sys_id from the Notify WebRTC Session table. **Note:** If you provide both a phone number and user sys_id, the method only uses the phone number.

Returns

Type	Description
void	

Example

This example demonstrates passing a phone number as the function parameter.

```
notifyClient.call({
    phoneNumber: "+18001112223"
});
```

Example

This example demonstrates passing a user record sys_id as the function parameter.

```
notifyClient.call({
    userId: "6816f79cc0a8016401c5a33be04be441"
});
```

Example

This example shows a button click handler.

```
$j("#pickupCallBtn").on("click", function() {
        notifyClient.hangupCall();
});
```

Example

This example shows an event handler.

```
onConnect: function(status) {
  // webRTC receives a call connection event (incoming or 
outgoing).
  if (status == SNC.Notify.Status.OPEN) {
    setStatus(getTimeStamp() + " -- Successfully establish
ed call");
    showHangupButton();
  }
},
```

NotifyClient - destroy()

Kills the current Notify client, rendering it unusable.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

NotifyClient - forwardCall(Object argument)

Forwards an ongoing incoming or outgoing phone call to either a different phone number or a different WebRTC client.

Parameters

Name	Type	Description
argument	Object	JSON object that contains the necessary information for forwarding the call to either a phone number or a WebRTC client (user sys_id). You can obtain this sys_id from the Notify WebRTC Session table.

Returns

Type	Description
void	

Example

This example demonstrates forwarding a call to a different phone number. The dtmf attribute allows you to send DTMF dial tones to the receiving number.

```
var arg = {
    type: "number",
    id: "+17012345678",
    dtmf: "1234"
}
client.forwardCall(arg);
```

Example

This example demonstrates forwarding a call to a different Notify client.

```
var arg = {
    type: "userId",
    id: "6816f79cc0a8016401c5a33be04be441"
}
client.forwardCall(arg);
```

NotifyClient - getAvailableClients(Function callback)

Returns a list of clients available to accept calls.

This method excludes the current client from the list. The equivalent Notify-getAvailableClients() method does not filter any user.

Parameters

Name	Type	Description
callback	Function	Function to use to parse the list of clients. This function accepts a single parameter, an array of JSON objects with the following format: `[{ sys_id: "...", // user's sys_id name: "..." // user's name }]`

Returns

Type	Description
void	

NotifyClient - getParentId(String callId, Function callback)

Returns the parent call identifier for a specified call identifier, if one exists.

Depending on the telephony provider, there may be a delay before the parent call identifier is returned; therefore you must provide a callback function.

Parameters

Name	Type	Description
callId	String	Unique identifier of the call for which to return the parent call identifier.
callback	Function	Function that obtains the JSON object that contains either the parent call identifier or an error message if the identifier could not be obtained after several tries.

Returns

Type	Description
String	Parent call identifier.

Example

This example shows how to use this method to obtain the parent call identifier.

```
notifyClient.getParentId( callId, function(jsonObj) {} );
```

Example

This example shows the contents of the jsonObj parameter.

```
{
        parentId: "xyz",
        error: "msg"
}
```

NotifyClient - getStatus()

Returns the normalized status of the current call.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Current status of the call. The values returned by the telephony provider API are normalized by replacing the returned driver value with its equivalent value as defined in SNC.Notify.Status.

Example

This example shows how to obtain the status of the current Notify client.

```
clientStatus = notifyClient.getStatus();
```

NotifyClient - hangupCall()

End the current call.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

This example how to hang up a call.

```
$j("#pickupCallBtn").on("click", function() {
    notifyClient.hangupCall();
});
```

NotifyClient - init()

Initializes the client driver.

For example, when using the Twilio client, it invokes the method Twilio.Device.setup(). Call this method after the user has interacted with the page. This initialization process is asynchronous, therefore, you must provide an ONLINE event handler. This handler is called when the setup process is complete and the system is ready to take or make calls.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

This example shows how to initialize the Notify client.

```
$j(function() {
  notifyClient = new SNC.Notify.Client( notifyConfig );
  notifyClient.setCallerId( '+31858889170' );
  notifyClient.init();
});
```

NotifyClient - mute(Boolean muted)

Mute or unmute the current client.

Parameters

Name	Type	Description
muted	Boolean	Mutes or unmutes the current call. • false: (or any non-true value) unmutes the current call. • true: mutes the current call.

Returns

Type	Description
void	

Example

This example shows how to send mute the current call.

```
notifyClient.mute( "true" );
```

NotifyClient - pickupCall()

Answers and connects to an incoming call from a WebRTC client.

Call this method when there is a notification of an incoming call.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

Example

This example shows how to pickup a call.

```
$j("#pickupCallBtn").on("click", function() {
    notifyClient.pickupCall();
});
```

NotifyClient - sendDtmf(String digits)

Send one or more DTMF-valid digits over the current call.

Parameters

Name	Type	Description
digits	String	One or more DTMF-valid digits.

Returns

Type	Description
void	

Example

This example shows how to send DTMF signals to the current call.

```
notifyClient.SendDtmf( "1246AF" ) {} );
```

NotifyClient - setCallerId(String value, Function autoSelectVendorCallback)

Sets the caller ID for the current client session.

You can change or update the caller ID at any time however, the caller ID must belong to the same vendor.

Parameters

Name	Type	Description
value	String	Phone number to use to make and receive calls.
autoSelectVendorCallback	Function	**Optional.** initializeVendorClientLazily must be set to "true" in the constructor to use this function, otherwise an error is thrown. Name of the callback function to call once the vendor is automatically set for the specified phone number. With this option, the vendor does not need to be specified in the constructor (notifyConfig.vendor). Auto vendor selection is an asynchronous operation. Therefore, this callback is required to indicate when it is safe to call notifyConfig.init(), as this method requires that the vendor be set before it is called. In addition, you must also check if notifyConfig.vendor

Name	Type	Description
		has been set in the callback to ensure that a vendor has been specified.

Returns

Type	Description
void	

Example

This example shows how to set the caller ID. This example assumes that the vendor is set in the constructor.

```
$j(function() {
  notifyClient = new SNC.Notify.Client( notifyConfig );
  notifyClient.setCallerId( '+31858889170' );
  notifyClient.init();
});
```

NotifyClient - setClientAvailable(Boolean available)

Sets the availability of an active WebRTC client agent.

This type of availability is different than an agent being in a call. In this case, an active WebRTC client may be connected and not on a call, but may not want to receive calls.

Calling this method updates the **Available** field value on the Notify Client Connected Session [notify_client_session] record associated with this client session. You can get a list of available clients using the getAvailableClients() method.

Parameters

Name	Type	Description
available	boolean	Flag that indicates whether an active WebRTC client wants to receive calls. • false: client does not want to receive calls. • true: client does want to receive calls.

Returns

Type	Description
void	

NotifyOnTaskClient - Client

The NotifyOnTaskClient API provides methods for sending SMS messages or starting/managing a conference call for various telephony service providers, such as Zoom and WebEx.

Any UI can consume the NotifyOnTaskClient API by explicitly including the NotifyOnTaskClient UI script.

Using the NotifyOnTaskClient API you can:

- Start a conference call
- End a conference call
- Add participants
- Perform actions that are available through the telephony driver such as:

- mute/unmute participants
- remove participants from a conference call
- add participants to a conference call
- start a conference call
- end a conference call

The Notify (com.snc.notify) plugin requires a separate subscription. For additional information on activating the Notify plugin, see Activate Notify.

NotifyOnTaskClient - addParticipants(Object data)

Adds the specified participants to a specified conference call.

Parameters

Name	Type	Description
data	Object	Object that describes the conference call.
data.addToWorkNotes	Boolean	Flag that indicates whether to add information about the participants that were included in the conference call in the work notes field of the associated record. For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record in which to add the work notes.

Name	Type	Description
		Default: false
data.confId	String	Required. Sys ID of the conference call. The conference Sys ID is located in the Notify Conference Call [notify_conference_call] table.
data.items	Array	Required. Information for each participant to include in the conference call. Valid array values: • id: User Sys ID; located in the User [sys_User] table. • notifyParticipantId: Participant Sys ID; located in the Notify Participant [notify_participant] table. • phoneNumber: Phone number of the participant. If this value is passed in conjunction with either the id or notifyParticipantId, this value supersedes the phone numbers in the user/ participant record and is used to place the call.

Name	Type	Description
		• email: email address of the participant.
data.message	String	Message that is read aloud when a user answers the call, such as, "P1 incident has been created please login to instance."
data.serviceProvider	String	Required. Name of conference service provider, such as Zoom or Webex.
data.sysId	String	Sys ID of the source record to associate with the conference call. For example, if a conference call is held to discuss a specific incident or problem, put the Sys ID of the incident or problem record in this value. This Sys ID is stored in the Source column of the NotifyConference Call [notify_conference_call] table and can later be tracked. This parameter is used in conjunction with the data.Table, data.addToWorkNotes, and

Name	Type	Description
		allowMulticonference parameters. You should configure this value when the conference call is initially created through a "start" action. If required, you can also set this value through this method.
data.table	String	Table that contains the source record to associate with the conference call. A source record can be any record, such as an "incident" or "problem", that is the topic of discussion in the conference call. This table name is stored in the Table column of the NotifyConference Call [notify_conference_call] table and can be tracked. This parameter is used in conjunction with the data.sysId, data.addToWorkNotes, and allowMulticonference parameters. You should configure this value when

Name	Type	Description
		the conference call is initially created through a "start" action. If required, you can also set this value through this method.

Returns

Type	Description
Object	Results of the conference action. `<action>.status`: Status of the conference action. • Data type: Boolean • Valid values: • true: Conference action succeeded • false: Conference action failed `<action>.successMessages`: If status is true, success message(s), else empty. • Data type: Array of Strings `<action>.warnMessages`: If status is false, any warning messages thrown during processing. • Data type: Array of Strings `<action>.errorMessages`: If status is false, any error messages thrown during processing.

Type	Description
	• Data type: Array of Strings

Example

Example

```
function addToConferenceCall() {
    var data = NotifyOnTaskClient.getNotifyActionTemplate(
);
    data.serviceProvider = 'Telephony'; // e.g 'Zoom', 'We
bEx'
    data.confId = 'Active conference sysId';
    data.items.push({ id: 'userSysId' });
    data.items.push({ phoneNumber: '+917799555331' });
    data.items.push({ email: 'yln99518@gmail.com' });

    NotifyOnTaskClient.addParticipants(data).then(functio
n (result) {
        var joinActionResult = result[0];
        if(joinActionResult.status) {
            joinActionResult.successMessages.forEach(funct
ion(msg) {
                console.log(msg);
            });
            return;
        }

        joinActionResult.warnMessages.forEach(function(msg
) {
            console.warn(msg);
        });
        joinActionResult.errorMessages.forEach(function(ms
g) {
            console.error(msg);
        });
    }, function (errMsg) {
        console.log(errMsg);
    });
}
```

NotifyOnTaskClient - doConferenceAction(String action, Object data)

Performs the specified conference call action, such as starting/ending a conference call or joining, removing, muting, or unmuting participants from a conference call.

You can start a new conference call and add participants within a single call to this method or call the method multiple times to start the call and then manage participants separately. In addition, through the passed in data object, you can configure the method to:

- Save pointers in the conference call record to the specific record (source record), such as an incident or problem, that is the topic of discussion for the conference call.
- Allow/disallow multiple conference calls for a source record.
- Automatically log the participants that were in the conference call in the **Work notes** field of the source record.
- Have a message read aloud when a participant answers an outgoing call from the conference.

Parameters

Name	Type	Description
action	String	Defines the conference call action to perform. The following are the available conference call actions: • start: Starts the conference call identified in data.confId • end: Terminates the conference call identified in data.confId

Name	Type	Description
		• join: Adds the participant specified in the data.items array to the conference call identified in data.confId • multiJoin: Adds the participants specified in the data.items array to the conference call identified in data.confId • selfJoin: Adds the currently logged in user to the conference call (no entry in data.items required.) • kick: Removes the participant specified in the data.items array from the conference call identified in data.confId • multiKick: Removes the participants specified in the data.items array from the conference call identified in data.confId • mute: Mutes the participant specified in the

Name	Type	Description
		data.items array on the conference call identified in data.confId • multiMute: Mutes the participants specified in the data.items array on the conference call identified in data.confId • unmute: Unmutes the participant specified in the data.items array from the conference call identified in data.confId • multiUnmute: Unmutes the participants specified in the data.items array from the conference call identified in data.confId
data	Object	Object that describes the conference call.
data.addToWorkNotes	Boolean	Flag that indicates whether to add information about the participants that were included in the conference call in the work notes field of the associated record.

Name	Type	Description
		For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record in which to add the work notes. Default: false Actions for which this parameter is valid: • start • join • multiJoin • selfJoin
data.allowMulticonference	Boolean	Flag that indicates whether to allow multiple conference calls for a specific record at one time. For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record that is allowed to have multiple conference calls. Default: false

Name	Type	Description
		Actions for which this parameter is valid: • start
data.confId	String	Sys ID of the conference call. The conference Sys ID is located in the Notify Conference Call [notify_conference_call] table. Actions for which this parameter is required: • end • join • multiJoin • selfJoin **Note:** Participant actions such as mute, unmute, and kick do not require this parameter to be set as the method obtains this information from the Notify Conference Call Participant [notify_participant] table.

Name	Type	Description
data.fromNumber	String	Service provider number to call into for the conference call. Locate this value in the Number or Phone number column of the Notify Phone Number [notify_number] table. Actions for which this parameter is required: • start
data.isNewConference	Boolean	Flag that indicates whether this is a new or an existing conference call. Valid values: • true: New conference call • false: Existing conference call Default: false Actions for which this parameter is valid: • start
data.message	String	Message that is read aloud when a user answers the call, such as, "P1 incident has

Name	Type	Description
		been created please login to instance." Actions for which this parameter is valid: • start • join • multiJoin
data.items	Array	Information for each participant to include in the conference call. Valid array values: • id: Sys ID of user; located in the User [sys_User] table. Valid actions: join, multiJoin, start • notifyParticipantId: Sys ID of the Notify participant; located in the Notify Participant [notify_participant] table. Valid actions: join, kick, multiJoin, mute, start, unmute • phoneNumber: Phone number of the participant. If this value is passed in conjunction with

Name	Type	Description
		either the id or notifyParticipantId, this value supersedes the phone numbers in the user/ participant record and is used to place the call. Valid actions: join, multiJoin, start • email: Email address of the participant. Valid actions: join, multiJoin, start
data.serviceProvider	String	Required. Name of conference service provider, such as Zoom or Webex. Actions for which this parameter is required: • all
data.sysId	String	Sys ID of the source record to associate with the conference call. For example, if a conference call is held to discuss a specific incident or problem, put the Sys ID of the incident or problem record in this value.

Name	Type	Description
		This Sys ID is stored in the Source column of the NotifyConference Call [notify_conference_call] table and can later be tracked. This parameter is used in conjunction with the data.Table, data.addToWorkNotes, and allowMulticonference parameters. Actions for which this parameter is valid: • start
data.table	String	Table that contains the source record to associate with the conference call. A source record can be any record, such as an "incident" or "problem", that is the topic of discussion in the conference call. This table name is stored in the Table column of the NotifyConference Call [notify_conference_call] table and can be tracked.

Name	Type	Description
		This parameter is used in conjunction with the data.sysId, data.addToWorkNotes, and allowMulticonference parameters. Actions for which this parameter is valid: • start

Returns

Type	Description
Object	Results of the conference action. `<action>.status`: Status of the conference action. • Data type: Boolean • Valid values: ◦ true: Conference action succeeded ◦ false: Conference action failed `<action>.successMessages`: If status is true, success message(s), else empty. • Data type: Array of Strings `<action>.warnMessages`: If status is false, any warning messages thrown during processing.

Type	Description
	• Data type: Array of Strings `<action>.errorMessages`: If status is false, any error messages thrown during processing. • Data type: Array of Strings

Example

The following example shows how to create a function to call doConferenceAction() to manipulate the participants in a conference call by passing in the action and the participants.

```
/**
 *
 * @param {string} action - action to perform on the confe
rence object or participant object
 * @param {Array} participants;
 */
function doConferenceAction(action, participants) {
    var data = NotifyOnTaskClient.getNotifyActionTemplate(
);
    data.serviceProvider = 'Telephony'; // e.g 'Zoom', 'We
bEx'
    data.confId = 'Active conference sysId';
    data.items = participants;

    NotifyOnTaskClient.doConferenceAction(action, data).th
en(function (result) {
        var kickActionResult = result[0];
        if (kickActionResult.status)
            console.log(action + ' succeeded');
        else {
            kickActionResult.warnMessages.forEach(functio
n (msg) {
                console.warn(msg);
            });
            kickActionResult.errorMessages.forEach(functio
n (msg) {
                console.error(msg);
            });
        }
    }, function (errMsg) {
            console.log(errMsg)
    });
}

// kick participants

doConferenceAction('kick', [{notifyParticipantId: 'notifyP
articipantSysId'}]);

// kick multiple participants

doConferenceAction('multiKick',
    [{notifyParticipantId: 'notifyParticipantSysId'},
    {notifyParticipantId: 'notifyParticipantSysId'}]);

// Mute participants
doConferenceAction('mute', [{notifyParticipantId: 'notifyP
articipantSysId'}]);
doConferenceAction('mute', [{notifyParticipantId: 'notifyP
articipantSysId'}]);

doConferenceAction('multiMute',
    [{notifyParticipantId: 'notifyParticipantSysId'},
    {notifyParticipantId: 'notifyParticipantSysId'}]);

// self join to any confernece.
doConferenceAction('selfJoin', [{id: 'logged in userId'}])
;
```

NotifyOnTaskClient - endConference(Object data)

Terminates the specified conference call.

Parameters

Name	Type	Description
data	Object	Object that describes the conference call.
data.confId	String	Sys ID of the conference call.
data.serviceProvider	String	Required. Name of conference service provider, such as Zoom or Webex.

Returns

Type	Description
Object	Results of the conference action. `<action>.status`: Status of the conference action. • Data type: Boolean • Valid values: • true: Conference action succeeded • false: Conference action failed `<action>.successMessages`: If status is true, success message(s), else empty. • Data type: Array of Strings `<action>.warnMessages`: If status is false, any warning messages thrown during processing.

Type	Description
	• Data type: Array of Strings `<action>.errorMessages`: If status is false, any error messages thrown during processing. • Data type: Array of Strings

Example

Example

```
function endConferenceCall() {
    var data = NotifyOnTaskClient.getNotifyActionTemplate(
);
    data.serviceProvider = 'Telephony'; // e.g 'Zoom', 'We
bEx'
    data.confId = 'Active conference sysId';

    NotifyOnTaskClient.endConference(data).then(function (
result) {
        var endActionResult = result[0];
        if (endActionResult.status)
            console.log('Conference has been ended');
        else {
            endActionResult.warnMessages.forEach(function
(msg) {
                console.warn(msg);
            });
            endActionResult.errorMessages.forEach(functio
n (msg) {
                console.error(msg);
            });
        }
    }, function (errMsg) {
        console.log(errMsg);
    });
}
```

NotifyOnTaskClient - getNotifyActionTemplate()

Returns a JSON data template to use with the doConferenceAction() method. Using this template automatically structures the data object so that you don't have to manually create it.

Call this method prior to calling the doConferenceAction() method. For the desired conference call action, set the desired parameters within the template, and then pass the template in the doConferenceAction() call. For additional information on the valid parameters for each action, see doConferenceAction().

> **Note:** This is a helper method. You can also manually construct this object and pass it into the doConferenceAction() method and have the same outcome.

Parameters

Name	Type	Description
None		

Returns

Type	Description
data	Object that describes the conference call.
data.addToWorkNotes	Flag that indicates whether to add information about the participants that were included in the conference call in the work notes field of the associated record. For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record in which to add the work notes.

Type	Description
	Default: false Actions for which this parameter is valid: • start • join • multiJoin • selfJoin
data.allowMultconference	Flag that indicates whether to allow multiple conference calls for a specific record at one time. For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record that is allowed to have multiple conference calls. Default: false Actions for which this parameter is valid: • start
data.confId	Sys ID of the conference call. The conference Sys ID is located in the Notify Conference Call [notify_conference_call] table. Actions for which this parameter is required: • end

Type	Description
	• join • multiJoin • selfJoin **Note:** Participant actions such as mute, unmute, and kick do not require this parameter to be set as the method obtains this information from the Notify Conference Call Participant [notify_participant] table.
data.fromNumber	Service provider number to call into for the conference call. Locate this value in the Number or Phone number column of the Notify Phone Number [notify_number] table. Actions for which this parameter is required: • start
data.isNewConference	Flag that indicates whether this is a new or an existing conference call. Valid values: • true: New conference call • false: Existing conference call Default: false

Type	Description
	Actions for which this parameter is valid: • start
data.items	Information for each participant to include in the conference call. Valid array values: • id: Sys ID of user; located in the User [sys_User] table. Valid actions: join, multiJoin, start • notifyParticipantId: Sys ID of the Notify participant; located in the Notify Participant [notify_participant] table. Valid actions: join, kick, multiJoin, mute, start, unmute • phoneNumber: Phone number of the participant. If this value is passed in conjunction with either the id or notifyParticipantId, this value supersedes the phone numbers in the user/participant record and is used to place the call. Valid actions: join, multiJoin, start • email: Email address of the participant. Valid actions: join, multiJoin, start

Type	Description
data.message	Message that is read aloud when a user answers the call, such as, "P1 incident has been created please login to instance." Actions for which this parameter is valid: • start • join • multiJoin
data.serviceProvider	Required. Name of conference service provider, such as Zoom or Webex. Actions for which this parameter is required: • all
data.sysId	Sys ID of the source record to associate with the conference call. For example, if a conference call is held to discuss a specific incident or problem, put the Sys ID of the incident or problem record in this value. This Sys ID is stored in the Source column of the NotifyConference Call [notify_conference_call] table and can later be tracked. This parameter is used in conjunction with the data.Table,

Type	Description
	data.addToWorkNotes, and allowMulticonference parameters. Actions for which this parameter is valid: • start
data.table	Table that contains the source record to associate with the conference call. A source record can be any record, such as an "incident" or "problem", that is the topic of discussion in the conference call. This table name is stored in the Table column of the NotifyConference Call [notify_conference_call] table and can be tracked. This parameter is used in conjunction with the data.sysId, data.addToWorkNotes, and allowMulticonference parameters. Actions for which this parameter is valid: • start

Example

The following example shows how to call getNotifyActionTemplate() to obtain the data template necessary to define the actions for doConferenceAction().

```
/**
```

```
 *
 * @param {string} action - action to perform on the confe
rence object or participant object
 * @param {Array} participants;
 */
function doConferenceAction(action, participants) {
    var data = NotifyOnTaskClient.getNotifyActionTemplate(
);
    data.serviceProvider = 'Telephony'; // e.g 'Zoom', 'We
bEx'
    data.confId = 'Active conference sysId';
    data.items = participants;

    NotifyOnTaskClient.doConferenceAction(action, data).th
en(function (result) {
        var kickActionResult = result[0];
        if (kickActionResult.status)
            console.log(action + ' succeeded');
        else {
            kickActionResult.warnMessages.forEach(functio
n (msg) {
                console.warn(msg);
            });
            kickActionResult.errorMessages.forEach(functio
n (msg) {
                console.error(msg);
            });
        }
    }, function (errMsg) {
            console.log(errMsg)
    });
}

// kick participants

doConferenceAction('kick', [{notifyParticipantId: 'notifyP
articipantSysId'}]);

// kick multiple participants

doConferenceAction('multiKick',
    [{notifyParticipantId: 'notifyParticipantSysId'},
    {notifyParticipantId: 'notifyParticipantSysId'}]);
```

```
// Mute participants
doConferenceAction('mute', [{notifyParticipantId: 'notifyP
articipantSysId'}]);
doConferenceAction('mute', [{notifyParticipantId: 'notifyP
articipantSysId'}]);

doConferenceAction('multiMute',
    [{notifyParticipantId: 'notifyParticipantSysId'},
    {notifyParticipantId: 'notifyParticipantSysId'}]);

// self join to any confernece.
doConferenceAction('selfJoin', [{id: 'logged in userId'}])
;
```

NotifyOnTaskClient - start(Object data)

Starts a new conference call.

Parameters

Name	Type	Description
data	Object	Object that describes the conference call.
data.addToWorkNotes	Boolean	Optional.Flag that indicates whether to add information about the participants that were included in the conference call in the work notes field of the associated record. For this functionality to work, you must also specify values for the data.table and data.sysId parameters to identify the record in which to add the work notes.

Name	Type	Description
		Default: false
data.allowMulticonference	Boolean	Optional. Flag that indicates whether to allow multiple conference calls for a specific record at one time. For this functionality to work, you must also specify values in the data.table and data.sysId parameters. These parameters identify the record that is allowed to have multiple conference calls. Default: false
data.fromNumber	String	Required. Service provider number to call into for the conference call. Locate this value in the Number or Phone number column of the Notify Phone Number [notify_number] table.
data.items	Array	Optional. Information for each participant to include in the conference call.

Name	Type	Description
		Valid array values: • id: User Sys ID; located in the User [sys_User] table. • notifyParticipantId: Participant Sys ID; located in the Notify Participant [notify_participant] table. • phoneNumber: Phone number of the participant. If this value is passed in conjunction with either the id or notifyParticipantId, this value supersedes the phone numbers in the user/participant record and is used to place the call. • email: Email address of the participant.
data.message	String	Optional. Message that is read aloud when a user answers the call, such as, "P1 incident has been created please login to instance."
data.serviceProvider	String	Required. Name of conference service

Name	Type	Description
		provider, such as Zoom or Webex.
data.sysId	String	Optional. Sys ID of the source record to associate with the conference call. For example, if a conference call is held to discuss a specific incident or problem, put the Sys ID of the incident or problem record in this value. This Sys ID is stored in the Source column of the NotifyConference Call [notify_conference_call] table and can later be tracked. This parameter is used in conjunction with the data.Table, data.addToWorkNotes, and allowMulticonference parameters.
data.table	String	Optional. Table that contains the source record to associate with the conference call. A source record can be any record, such as an "incident" or "problem", that is the topic of discussion in the conference call.

Name	Type	Description
		This table name is stored in the Table column of the NotifyConference Call [notify_conference_call] table and can be tracked. This parameter is used in conjunction with the data.sysId, data.addToWorkNotes, and allowMulticonference parameters.

Returns

Type	Description
Object	Results of the conference action. `<action>.status`: Status of the conference action. • Data type: Boolean • Valid values: ◦ true: Conference action succeeded ◦ false: Conference action failed `<action>.successMessages`: If status is true, success message(s), else empty. • Data type: Array of Strings

Type	Description
	`<action>.warnMessages`: If status is false, any warning messages thrown during processing. • Data type: Array of Strings `<action>.errorMessages`: If status is false, any error messages thrown during processing. • Data type: Array of Strings

Example

Example

```
function startConferenceCall() {
    var data = NotifyOnTaskClient.getNotifyActionTemplate(
);
    data.table = 'incident';
    data.sysId = '1234';
    data.serviceProvider = serviceProvider;
    data.addToWorkNotes = true;
    data.fromNumber = 'Telephony Number';
    data.items.push({ id: 'userSysId' });
    data.items.push({ phoneNumber: '+917799555332' });
    data.items.push({ email: 'yln99517@gmail.com' });

    NotifyOnTaskClient.start(data).then(function (result)
{
        var startActionResult = result[0];
        if(startActionResult.status) {
            startActionResult.successMessages.forEach(func
tion(msg) {
                console.log(msg);
            });
            return;
        }

        startActionResult.warnMessages.forEach(function(ms
g) {
            console.warn(msg);
        });
        startActionResult.errorMessages.forEach(function(m
sg) {
            console.error(msg);
        });
    }, function (errMsg) {
        console.log(errMsg);
    });
}
```

ScopedSessionDomain - Client

The ScopedSessionDomain API is a script include that contains client-side methods that provide functionality related to the current session domain.

This API is only available if the Domain Support - Domain Extensions Installer plugin (com.glide.domain.msp_extensions) plugin has been activated in the instance. In addition, the caller must have the admin role to access this API.

ScopedSessionDomain - getCurrentDomainID()

Returns the sys_id of the current domain for the logged-in user session.

The identifier that is returned depends on the domain type and the instantiation of that domain.

- If the user is configured in the global domain, and does not use the domain picker to switch domains, the method returns null.
- If the user uses the domain picker to switch to the global domain, the method returns the string "global".
- For all other domains, the method returns the sys_id of that domain.

To access this method from a client-side script, you must use GlideAjax() to invoke it. To invoke this method from a server-side script, use something similar to the following to instantiate the object and access the method.

```
var ssg = new global.ScopedSessionDomain();
domainID = ssg.getCurrentDomainID();
```

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	Sys_id of the session domain of the current logged-in user. This is the same information that appears in the domain picker.

Example

This example shows how to call the getCurrentDomainID() method from a client-side script.

```
// This example is calling the script include in a client
script.
// This particular record is within the "Service Portal -
Standard Ticket" scope.
// To reproduce this example:
//  1. Change application scope to "Service Portal - Stand
ard Ticket"
//  2. Navigate to Client Script table and open a new form
 and name it anything
//  3. Set table=ticket_configuration (this table is withi
n the same scope)
//  4. Set UI type=Desktop and Type=onLoad
//  5. Populate the script field with the script above
//  6. Navigate to the ticket_configuration table and open
 any form.
//
// This will trigger the client script, invoke the API, an
d pop up a browser alert containing the sys_id of the user
's current domain

function onLoad() {
  var ga = new GlideAjax("global.ScopedSessionDomain"); //
 Set the script include
  ga.addParam("sysparm_name", "getCurrentDomainID"); // Se
t the getCurrentDomainID method
  ga.getXML(getResponse);

  function getResponse(response) {
     var answer = response.responseXML.documentElement.get
Attribute('answer');
     alert(answer); // Pops up the sys_id of the domain re
cord
  }
}
```

ScriptLoader - Client

Provides the ability to load scripts asynchronously.

You can use the ScriptLoader API in client-side scripts for a platform/ desktop UI using ListV2 and ListV3 APIs. It is not available for Service Portal, Mobile, or Agent Workspace.

You access the ScriptLoader methods by using the global object ScriptLoader.

ScriptLoader - getScripts(Array scripts, Function callback)

Loads scripts asynchronously.

Parameters

Name	Type	Description
scripts	Array	Array of scripts to load.
callback	Function	Function to call when the scripts have been loaded. The callback function must not have any arguments.

Returns

Type	Description
void	

Example

```
var scriptFiles=[
'scripts/classes/providers/ProviderUI.js',
'scripts/classes/providers/wf_provider_client_script.js'
];

ScriptLoader.getScripts(scriptFiles, function() {
  ProviderUI.setFields(['input_transform']);
  ProviderUI.apply();
  ProviderUI.removeDisabledAttribute("sys_readonly.wf_elem
ent_activity.access");
}
```

ScriptLoader - getScripts(String filePath, Function callback)

Gets scripts asynchronously.

Parameters

Name	Type	Description
filePath	String	Path, including the file name, that contains one or more scripts.
callback	Function	Function to call after the scripts have been loaded. This callback function should not have arguments.

Returns

Type	Description
void	

Example

This example shows how load a utility script and then run the getDepartment() function.

```
// Client script to load a utility script and run the getD
epartment callback function
ScriptLoader.getScripts('sn_ui_script_util.Utilities.jsdbx
', getDepartment);

function getDepartment() {
  var req = sn_ui_script_util.Utilities.rest('json');
  req.addParam("sysparm_query", "sys_id=" + newValue);
  req.addParam("sysparm_fields", "department");
  req.addParam("sysparm_display_value", true);
  req.success(updateNotes);
  req.get("/api/now/table/sys_user");
}

function updateNotes(data) {
  g_form.setValue("work_notes", data.result[0].department.
display_value);
}
```

SNAnalytics - Client

The SNAnalytics API provides methods to push custom analytics data (events, pages, and user properties) to the User Experience Analytics for Service Portal dashboard.

User Experience Analytics for Service Portal provides dashboard views for monitoring the key performance indicators (KPIs) of web applications built on Service Portal. You can use these insights to optimize your portal. For example, User Experience Analytics tracks when a user orders a catalog item or views a knowledge article. You can use this data to infer which items or articles are the most popular among users.

To access this API, the Service Portal Analytics (com.glide.service-portal.analytics) plugin must be activated on the instance. In addition, within your application, you must import the snAnalytics Angular service, such as:

```
<client_script><![CDATA[function($rootScope, $scope, $wind
ow, $timeout, spUtil, $sce, spModal, $uibModal, $location
, cabrillo, snAnalytics)
```

For additional information, see User Experience Analytics for Service Portal.

SNAnalytics - addEvent(Object payload)

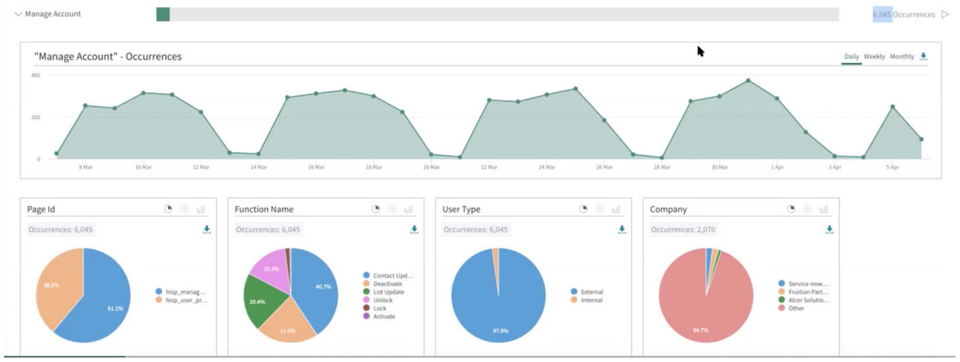
Stores the specified event information in the analytics data store.

Events are actions performed by a user, such as clicking a button or submitting a form. Call this method within your web-page widget whenever you want to capture a user action. These events then automatically appear on the associated user session timeline and User Experience Analytics dashboard.

The following is an example of a payload passed in an addEvent() call:

```
var payload= {};
payload.name = "Manage Account";
payload.data = {};
payload.data["Function Name"] = c.data.function_name;
payload.data["User Type"] = c.data.user_type;
payload.data["Company"] = c.data.company_name;
snAnalytics.addEvent(payload);
```

The screen capture below shows the information that appears on the analytics dashboard for the event "Manage Account." The timeline at the top represents the number of times that the event occurred. The pie charts below the timeline reflect the properties that were captured in the addEvent() call.



If you want to capture when users access a web page, use the SNAnalytics - startPage(String name, String description) method instead of this method.

Parameters

Name	Type	Description

payload	Object	Event to store in the analytics data store.
		
		Data type: Object
		
		`"payload": {`
		`  "data": [Array],`
		`  "name": String`
		`}`

payload.data	Array Each element can be a string (case-sensitive), boolean, number, or date.	Optional. Name-value pairs of custom event properties. These properties can be any values that you want to track and see on the analytics dashboard. They appear under the associated event timeline on the analytics dashboard. The **Page Id** property always appears first on the dashboard for all base system events, and all other properties are sorted alphabetically. If no properties are required for an event, only an event timeline appears on the analytics

Name	Type	Description
		dashboard. Properties can be added at a later time. The following values are automatically converted by this method: • String value of "yes": Boolean value of "true" • String value of "no": Boolean value of "false Default value: Null
payload.name	String	Descriptive name of the event. Special characters are not allowed. Maximum length: The length of the event name and value cannot exceed 300 bytes.

Returns

Type	Description
None	

Example

The following example shows how to call the addEvent() method during initialization of a widget.

```
function initialize() {
  c.options.glyph = c.options.glyph || 'search';
  c.options.title = c.options.title || c.data.searchMsg;
  c.options.color = c.options.color || "default";
  c.searchTerm = c.data.q;
  c.searchQuery = "";
  c.pageID = $scope.page && $scope.page.id;
  c.showSuggestions =  c.data.searchTypeBehavior === "sugg
estions" && c.data.isSuggestionsEnabled === "true";
  c.suggestionsLimit = c.options.limit || "";
  c.latitude = null;
  c.longitude = null;
  c.isLocationTrackerDisabled = c.data.isLocationTrackerDi
sabled === "true";
  c.isTypeAheadEnabled = c.data.isTypeAheadEnabled === "tr
ue";

  c.sendAnalytics = function(type){
    var payload= {};
    payload.name = "Initiate Search";
    payload.data = {};
    payload.data["Keyword"] = (type == 'User Entered' ? c.
searchTerm : c.searchQuery);
    payload.data["Type"] = type;
    snAnalytics.addEvent(payload);
  };
}
```

SNAnalytics - appendToUserProperty(String name, String value)

Appends the specified string to the specified user string property in the analytics data store.

Parameters

Name	Type	Description
name	String or String[]	Name of the property to append the specified string to. Special characters are not allowed. **Note:** The associated property must be a string or string[]. Maximum length: The length of the property name and property value cannot exceed 300 bytes.
value	String	Value to append to the string property. The following values are automatically converted by this method: • String value of "yes": Boolean value of "true"

Name	Type	Description
		• String value of "no": Boolean value of "false

Returns

Type	Description
None	

Example

This example shows how to add `television` to the `tags` property.

```
snAnalytics.setUserProperties({
  level: 7,
  lastPurchase: new Date(),
  lastPurchaseId: '41563cd2-1666-4855-8c0d-b9ca778aed23',
  isPremium: true,
  tags: ['chair', 'table'],
});

// Append television to the tags property (now 'tags' wil
l have 'chair', 'table', and 'television')
snAnalytics.appendToUserProperty('tags', 'television');
```

SNAnalytics - incUserProperty(String name, Number value)

Increments or decrements the specified user property value with the specified number value in the analytics data store.

Parameters

Name	Type	Description
name	String	Name of the property to increment. Value is case-sensitive. **Note:** The associated property must be a number.

Name	Type	Description
value	Number	Amount to increment the property by. If you enter a negative number, the value is decremented.

Returns

Type	Description
None	

Example

The following example shows how to increment the property `Grace days` by 5.

```
snAnalytics.incUserProperty('Grace days', 5)
```

SNAnalytics - removeUserProperty(String name)

Removes the specified property for the current user from the analytics data store.

In addition, the property no longer appears on the analytics dashboard.

Parameters

Name	Type	Description
name	String	Name of the property to remove. Value is case-sensitive.

Returns

Type	Description
None	

Example

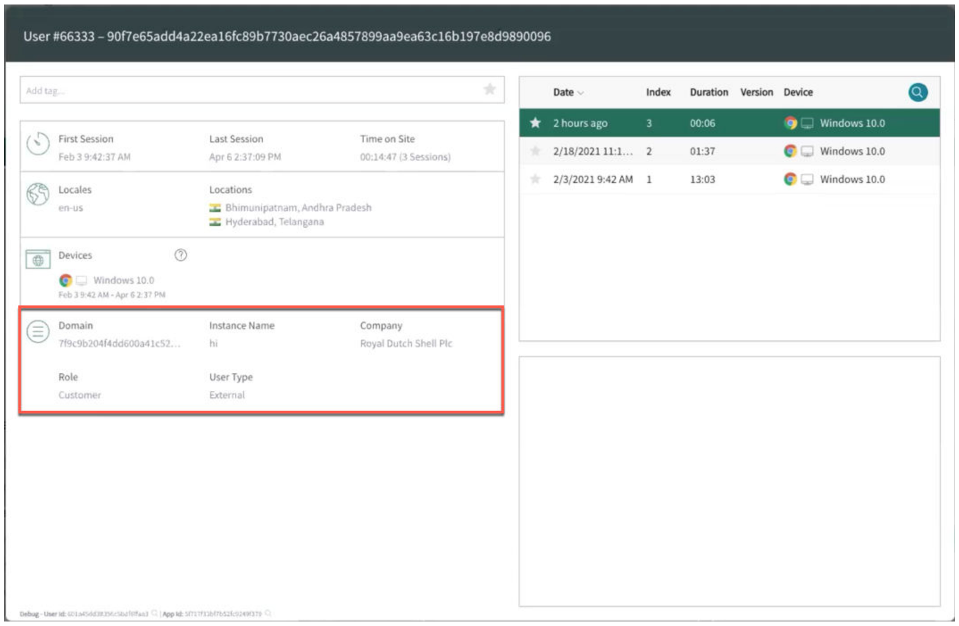
The following example shows how to remove the IsAdmin property.

```
snAnalytics.removeUserProperty('IsAdmin');
```

SNAnalytics - setUserProperties(Object properties)

Sets the specified properties with the specified values for the current user in the analytics data store.

These properties are saved in the analytics data store and appear on the user session details page as illustrated below. If a property already exists in the analytics data store, the current value is overwritten with the new value.


User #66333 – 90f7e65add4a22ea16fc89b7730aec26a4857899aa9ea63c16b197e8d9890096

First Session Feb 3 9:42:37 AM	Last Session Apr 6 2:37:09 PM	Time on Site 00:14:47 (3 Sessions)
Locales en-us	Locations Bhimunipatnam, Andhra Pradesh Hyderabad, Telangana	
Devices Windows 10.0 Feb 3 9:42 AM - Apr 6 2:37 PM		
Domain 7f9c9b204f4dd600a41c52...	Instance Name hi	Company Royal Dutch Shell Plc
Role Customer	User Type External	

	Date	Index	Duration	Version	Device
★	2 hours ago	3	00:06		Windows 10.0
☆	2/18/2021 11:1...	2	01:37		Windows 10.0
☆	2/3/2021 9:42 AM	1	13:03		Windows 10.0

Parameters

Name	Type	Description

properties	Object
	
	Each element in this object can be a string,
		Object that contains the name-value pairs of the user properties to set, such as:
		`{`
		`  level: 7,`
		`  lastPurchase: new Date(),`
		`  lastPurchaseId: '41563cd2-1666-4855`

Name	Type	Description

	boolean, number, date, string[], or null.	`-8c0d-b9ca778aed23',`
		`  isPremium: true,`
		`  tags: ['chair', 'table'],`
		`}`
		The following values are automatically converted by this method:
		• String value of "yes": Boolean value of "true"
		• String value of "no": Boolean value of "false

Returns

Type	Description
None	

Example

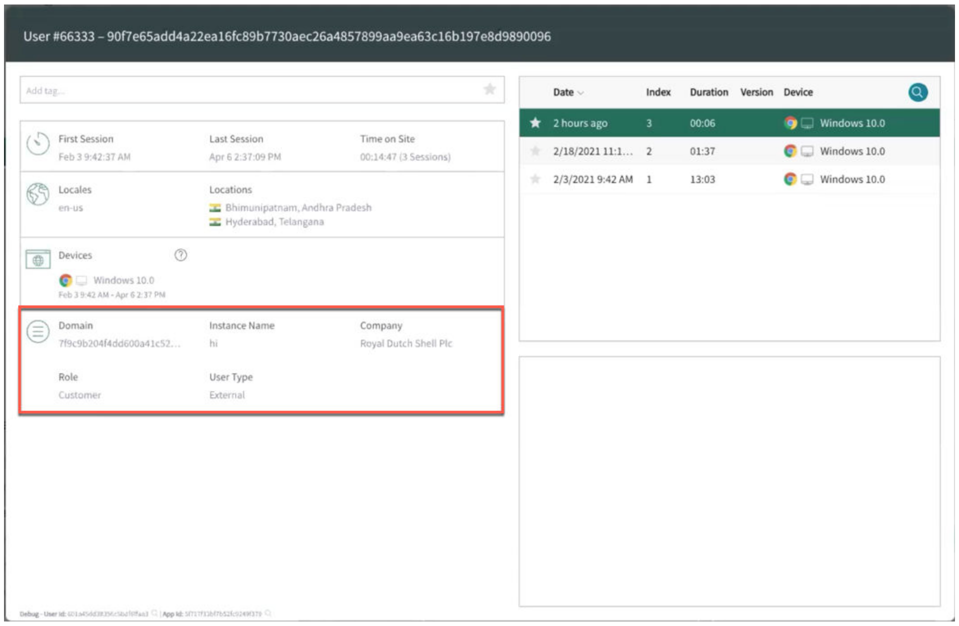
The following example shows how to set multiple properties for the current user.

```
snAnalytics.setUserProperties({
  level: 7,
  lastPurchase: new Date(),
  lastPurchaseId: '41563cd2-1666-4855-8c0d-b9ca778aed23',
  isPremium: true,
  tags: ['chair', 'table'],
});
```

SNAnalytics - setUserProperty(String name, UserProperty value)

Sets the specified property with the specified value for the current user in the analytics data store.

These properties are saved in the analytics data store and appear on the user session details page as illustrated below. If a property already exists in the analytics data store, the current value is overwritten with the new value.



User #66333 – 90f7e65add4a22ea16fc89b7730aec26a4857899aa9ea63c16b197e8d9890096

Add tag...

First Session	Last Session	Time on Site
Feb 3 9:42:37 AM	Apr 6 2:37:09 PM	00:14:47 (3 Sessions)

Locales: en-us

Locations: Bhimunipatnam, Andhra Pradesh; Hyderabad, Telangana

Devices: Windows 10.0, Feb 3 9:42 AM - Apr 6 2:37 PM

Domain	Instance Name	Company
7f9c9b204f4dd600a41c52...	hi	Royal Dutch Shell Plc

Role	User Type
Customer	External

Date	Index	Duration	Version	Device
2 hours ago	3	00:06		Windows 10.0
2/18/2021 11:1...	2	01:37		Windows 10.0
2/3/2021 9:42 AM	1	13:03		Windows 10.0

Parameters

Name	Type	Description
name	String	Name of the property to update. This name appears as the label for the property. For example, in the prior screenshot, Domain, Instance Name, Company, Role, and User Type are all name parameters. Special characters are not allowed. Maximum length: The length of the property name and property value cannot exceed 300 bytes.
value	UserProperty This value can be a string, boolean,	Value to set in the specified property. The following values are automatically converted by this method: • String value of "yes": Boolean value of "true" • String value of "no": Boolean value of "false

Name	Type	Description
	number, date, string[], or null.	

Returns

Type	Description
None	

Example

The following example shows how to set the property `Company`.

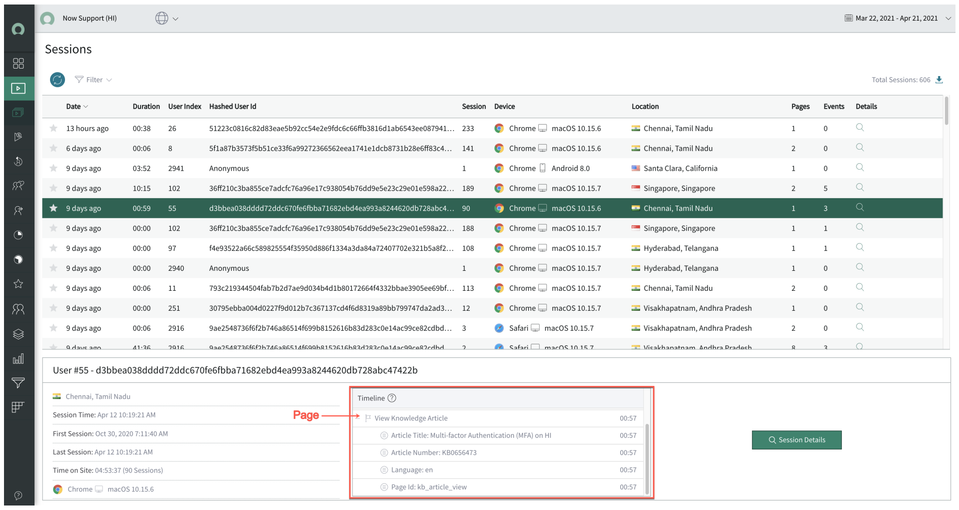
```
snAnalytics.setUserProperty('Company', "ABC Company")
```

SNAnalytics - startPage(String name, String description)

Saves the name and description of a page in the analytics data store.

This information appears in the user session timeline and on the analytics dashboard. Call this method within your custom widgets to track the pages visited by a user. You can also use this method to track user navigation within an individual page. For more information, see Use User Experience Analytics.

Note: In general, portal pages are automatically tagged with this tracking capability. Use this method for custom scenarios, such as a single page custom widget in a wizard scenario.



Sessions

Total Sessions: 606

	Date	Duration	User Index	Hashed User Id	Session	Device		Location	Pages	Events	Details
	13 hours ago	00:38	26	51223c0816c82d83eae5b92cc54e2e9fdc6c66ffb3816d1ab6543ee087941…	233	Chrome	macOS 10.15.6	Chennai, Tamil Nadu	1	0	
	6 days ago	00:06	8	5f1a87b3573f5b51ce33f6a99272366562eea1741e1dcb8731b28e6ff83c4…	141	Chrome	macOS 10.15.6	Chennai, Tamil Nadu	2	0	
	9 days ago	03:52	2941	Anonymous	1	Chrome	Android 8.0	Santa Clara, California	1	0	
	9 days ago	10:15	102	36ff210c3ba855ce7adcfc76a96e17c938054b76dd9e5e23c29e01e598a22…	189	Chrome	macOS 10.15.7	Singapore, Singapore	2	5	
	9 days ago	00:59	55	d3bbea038dddd72ddc670fe6fbba71682ebd4ea993a8244620db728abc4…	90	Chrome	macOS 10.15.6	Chennai, Tamil Nadu	1	3	
	9 days ago	00:00	102	36ff210c3ba855ce7adcfc76a96e17c938054b76dd9e5e23c29e01e598a22…	188	Chrome	macOS 10.15.7	Singapore, Singapore	1	1	
	9 days ago	00:00	97	f4e93522a66c589825554f35950d886f1334a3da84a72407702e321b5a8f2…	108	Chrome	macOS 10.15.7	Hyderabad, Telangana	1	1	
	9 days ago	00:00	2940	Anonymous	1	Chrome	macOS 10.15.7	Hyderabad, Telangana	1	0	
	9 days ago	00:06	11	793c219344504fab7b2d7ae9d034b4d1b80172664f4332bbae3905ee69bf…	113	Chrome	macOS 10.15.7	Chennai, Tamil Nadu	2	0	
	9 days ago	00:00	251	30795ebba004d0227f9d012b7c367137cd4f6d8319a89bb799747da2ad3…	12	Chrome	macOS 10.15.7	Visakhapatnam, Andhra Pradesh	1	0	
	9 days ago	00:06	2916	9ae2548736f6f2b746a86514f699b8152616b83d283c0e14ac99ce82cdbd…	3	Safari	macOS 10.15.7	Visakhapatnam, Andhra Pradesh	2	0	
	9 days ago	41:36	2916	9ae2548736f6f2b746a86514f699b8152616b83d283c0e14ac99ce82cdbd	2	Safari	macOS 10.15.7	Visakhapatnam, Andhra Pradesh	8	3	

User #55 - d3bbea038dddd72ddc670fe6fbba71682ebd4ea993a8244620db728abc47422b

Chennai, Tamil Nadu

Session Time: Apr 12 10:19:21 AM

First Session: Oct 30, 2020 7:11:40 AM

Last Session: Apr 12 10:19:21 AM

Time on Site: 04:53:37 (90 Sessions)

Chrome macOS 10.15.6

Timeline

View Knowledge Article	00:57
Article Title: Multi-factor Authentication (MFA) on HI	00:57
Article Number: KB0656473	00:57
Language: en	00:57
Page Id: kb_article_view	00:57

Parameters

Name	Type	Description
name	String	Descriptive name of the page or page section. Special characters are not allowed.
description	String	Optional. Description of the page to appear in the timeline and analytics dashboard. Default: name parameter value

Returns

Type	Description
None	

Example

The following example shows how to call the startPage() method.

```
snAnalytics.startPage('login_view', 'Login');
```

spAriaUtil - Client

Show messages on a screen reader.

spAriaUtil is an angular service that you can use in Service Portal widget client scripts.

spAriaUtil - sendLiveMessage(String message)

Announce a message to a screen reader.

The sendLiveMessage() method injects text into an aria-live region on the page. The default setting for an aria-live region is `assertive`, which means that messages are announced immediately. This can annoy and confuse users if used too frequently.

Parameters

Name	Type	Description
message	String	The message to be shown.

Returns

Type	Description
void	

Example

```
function(spAriaUtil) {
  /* widget controller */

  spAriaUtil.sendLiveMessage('Hello world!');
}
```

spContextManager - Client

Make data from a Service Portal widget available to other applications and services in a Service Portal page. For example, pass widget data to Agent Chat when it opens in a Service Portal page.

spContextManager is an AngularJS service that you can use in Service Portal widget client scripts.

Keys passed to this API are unique per page. For example, if the `'agent-chat'` key is already initialized by another widget on the page through the addContext() method, you must use the updateContextForKey() method to update the key's data.

For more information about passing data to Agent Chat, see Configure Agent Chat in Service Portal.

spContextManager - addContext(String key, Object context)

Initializes a key and adds widget data as the value. For example, add data to the 'agent-chat' key to make it available to Agent Chat.

Use this method the first time data is added to a specific key on a Service Portal page. If the key is already used by another widget on the page, use the updateContextForKey() method instead.

Parameters

Name	Type	Description
key	String	Name of the key to send the data. Available keys include**agent-chat**: Sends widget data to Agent Chat when it opens in a Service Portal page.
context	Object	Widget data in JSON format to send to the application or service specified in the key parameter. For example, `{'approval_count': 5}`.

Returns

Type	Description
void	

Example

Pass `approval_count` to Agent Chat. When a user initiates an Agent Chat conversation from the Service Portal homepage, the system appends `&sysparm_approval_count=5` to the Agent Chat `iframe` URL.

```
function ($scope, spContextManager) {
    spContextManager.addContext('agent-chat', {
        'approval_count': 5
    });
};
```

spContextManager - getContext()

Returns each key and associated data object defined by any widget on the page.

Using this method may affect performance. Use this method to understand which keys are initialized on the page and to get their current values. If you know which key you need to access, use the getContextForKey() method instead.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Object	Each key and associated data object defined on the page.

Example

```
function ($scope, spContextManager) {
  spContextManager.getContext();
}
```

spContextManager - getContextForKey(String key, Boolean returnPromise)

Returns the widget data associated with a key.

Parameters

Name	Type	Description
key	String	Name of the key to get context from.
returnPromise	Boolean	Flag that determines whether to return the data associated with a key as a promise or an object. Values include: • True: return the data as a promise. Use this option if another widget on the page uses the addContext() method to initialize the same key. Returning a promise prevents returning an undefined object when the key is not yet initialized. • False: returns an object containing

Name	Type	Description
		the data associated with the key.

Returns

Type	Description
Promise	If returnPromise is true, returns a promise that is fulfilled when another widget on the page initializes the key.
Object	If returnPromise is false, returns an object containing the data associated with the key. For example, `{approval_count: 5}`.

Example

Pass `approval_count` to Agent Chat. When a user initiates an Agent Chat conversation from the Service Portal homepage, the system appends `&sysparm_approval_count=5` to the Agent Chat `iframe` URL.

```
function ($scope, spContextManager) {
  spContextManager.getContextForKey('agent-chat', true).th
en(function(context) {
    context = context || {};
    context.approval_count = 5;
    spContextManager.updateContextForKey('agent-chat', con
text);
  });
}
```

spContextManager - updateContextForKey(String key, Object context)

Sends data to an existing key. For example, if another widget on the page uses the 'agent-chat' key to pass data to the Agent Chat configuration, you must update the context of the key rather than using the addContext() method.

Parameters

Name	Type	Description
key	String	Name of the key to send the data. Available keys include**agent-chat**: Sends widget data to Agent Chat when it opens in a Service Portal page.
context	Object	Widget data in JSON format to send to the application or service specified in the key parameter. For example, `{'approval_count': 5}`.

Returns

Type	Description
void	

Example

Pass `approval_count` to Agent Chat. When a user initiates an Agent Chat conversation from the Service Portal homepage, the system appends `&sysparm_approval_count=5` to the Agent Chat `iframe` URL.

```
function ($scope, spContextManager) {
  spContextManager.getContextForKey('agent-chat', true).th
en(function(context) {
    context = context || {};
    context.approval_count = 5;
    spContextManager.updateContextForKey('agent-chat', con
text);
  });
}
```

spModal - Client

Show alerts, prompts, and confirmation dialogs in Service Portal widgets. The SPModal class is available in Service Portal client scripts.

You can use spModal.open() to display a widget in a modal dialog. The spModal class is a lightweight wrapper for Angular UI bootstrap's $uibModal.

spModal - alert(String message).then(fn)

Displays an alert.

Parameters

Name	Type	Description
message	String	The message to display.

Returns

Type	Description
Boolean	The promise contains a single argument that contains true or false.

Example

```
// HTML template
<button ng-click="c.onAlert()" class="btn btn-default">
    Alert
  </button>

// Client script
function(spModal) {
    var c = this;
  c.onAlert = function () {
        spModal.alert('How do you feel today?').then(funct
ion (answer) {
            c.simple = answer;
        });
    }
 }
```

spModal - confirm(String message).then(fn)

Displays a confirmation message.

Parameters

Name	Type	Description
message	String	The message to display.

Returns

Type	Description
Boolean	The promise contains a single argument that contains true or false.

Example

```
// HTML template
 <button ng-click="c.onConfirm()" class="btn btn-default"
> Confirm </button>
<span>{{c.confirmed}}</span>

// Client script
function(spModal) {
  var c = this;
  c.onConfirm = function() {
        c.confirmed = "asking";
        spModal.confirm("Can you confirm or deny this?").t
hen(function(confirmed) {
            c.confirmed = confirmed; // true or false
        })
```

```
    }
 }
```

Example

Confirm with HTML message:

```
//HTML template
<button ng-click="c.onConfirmEx()" class="btn btn-default"
>
    Confirm - HTML message
  </button>
  <span>{{c.confirmed}}</span>

// Client script
function(spModal) {
  var c = this;
  // more control, passing options
    c.onConfirmEx = function() {
        c.confirmed = "asking";
        var warn = '<i class="fa fa-warning" aria-hidden="
true"></i>';
        spModal.open({
            title: 'Delete this Thing?',
            message: warn + ' Are you <b>sure</b> you wan
t to do that?'
        }).then(function(confirmed) {
            c.confirmed = confirmed;
        })
    }
}
```

spModal - open(Object options).then(fn)

Opens a modal window using the specified options.

Parameters

Name	Type	Description
options	Object	An object that may have these properties.

Name	Type	Description
		• title - a string that can be HTML that goes in the header. The default is empty. • message - a string that can be HTML that goes in the header. The default is empty. • buttons - an array that contains the buttons to show on the dialog. The default is Cancel and OK. • input - a Boolean. When true shows an input field on the dialog. The default is false. • value - a string containing the value of the input field. The default is empty. • widget - a string that identifies the widget ID or sys_id to embed in the dialog. The default is empty. • widgetInput - an object to send the embedded widget as input. The default is null. • shared - a client-side object to share data

Name	Type	Description
		with the embedded widget client script. • size - a string indicating the size of the window. Can be 'sm' or 'lg'. The default is empty.

Returns

Type	Description
void	

Example

Example of a prompt with a label

```
//HTML template
<button ng-click="c.onOpen()" class="btn btn-default">
    Prompt with label
  </button>
  <div ng-show="c.name">
    You answered <span>{{c.name}}</span>
  </div>

//Client code
function(spModal) {
  var c = this;
  c.onOpen = function() {
        //ask the user for a string
        spModal.open({
            title: 'Give me a name',
            message: 'What would you like to name it?',
            input: true,
            value: c.name
        }).then(function(name) {
            c.name = name;
        })
    }
}
```

Example

Example of agree with custom buttons.

```
//HTML template
<button ng-click="c.onAgree()" class="btn btn-default">
    Agree
  </button>
  <div ng-show="c.agree">
    You answered {{c.agree}}
  </div>

//Client script
function(spModal) {
  var c = this;
  c.onAgree = function() {
        // ask the user for a string
        // note embedded html in message
        var h = '<h4>Apple likes people to agree to lots o
f stuff</h4>'
        // Line feeds added to the following lines for pre
sentation formatting.
        var m = 'Your use of Apple software or hardware pr
oducts is based
on the software license and other terms and conditions in
effect for the
product at the time of purchase. Your agreement to these t
erms is required
to install or use the product. '
        spModal.open({
            title: 'Do you agree?',
            message: h + m,
            buttons: [
                {label:'□ ${No}', cancel: true},
                {label:'□ ${Yes}', primary: true}
            ]
        }).then(function() {
            c.agree = 'yes';
        }, function() {
            c.agree = 'no';
        })
    }
}
```

Example

Example of embedded widget

```
//HTML template
<button ng-click="c.onWidget('widget-cool-clock')" class="
btn btn-default">
    Cool Clock
  </button>

//Client script
function(spModal) {
  var c = this;
  c.onWidget = function(widgetId, widgetInput) {
        spModal.open({
            title: 'Displaying widget ' + widgetId,
            widget: widgetId,
            widgetInput: widgetInput || {}
        }).then(function(){
            console.log('widget dismissed');
        })
    }
}
```

spModal - prompt(String message, String default).then(fn)

Displays a prompt for user input.

Parameters

Name	Type	Description
message	String	The message to display.
default (optional)	String	A default value to use if the user does not provide a response.

Returns

Type	Description
String	The promise contains the user's response, or the default value if the user does not enter a response.

Example

```
//HTML template
 <button ng-click="c.onPrompt()" class="btn btn-default">
    Prompt
  </button>
  <div ng-show="c.name">
    You answered <span>{{c.name}}</span>
  </div>

// Client script
function(spModal) {
  var c = this;
  c.onPrompt = function() {
        spModal.prompt("Your name please", c.name).then(fu
nction(name) {
            c.name = name;
        })
    }
}
```

spUtil - Client

Utility methods to perform common functions in a Service Portal widget client script.

spUtil - addErrorMessage(String message)

Displays a notification error message.

Parameters

Name	Type	Description
message	String	Error message to display.

Returns

Type	Description
void	

Example

```
spUtil.addErrorMessage("There has been an error processin
g your request")
```

spUtil - addInfoMessage(String message)

Displays a notification info message.

Parameters

Name	Type	Description
message	String	Message to display.

Returns

Type	Description
void	

Example

```
spUtil.addInfoMessage("Your order has been placed")
```

spUtil - addTrivialMessage(String message)

Displays a trivial notification message.

Trivial messages disappear after a short period of time.

Parameters

Name	Type	Description
message	String	Message to display.

Returns

Type	Description
void	

Example

```
spUtil.addTrivialMessage("Thanks for your order")
```

spUtil - createUid()

Create a unique ID.

Parameters

Name	Type	Description
None		

Returns

Type	Description
string	A unique 32 character id.

spUtil - get(String widgetId Object data)

Returns a widget model by ID or sys_id.

Use this method to embed a widget model in a widget client script. The callback function returns the full widget model.

Parameters

Name	Type	Description
widgetId	String	Widget ID or sys_id of the widget to embed.
data	Object	(Optional) Name/ value pairs of parameters to pass to the widget model.

Returns

Type	Description
Object	Model of the embedded widget.

Example

Without data passed

```
spUtil.get("widget-cool-clock").then(function(response) {
  c.coolClock = response;
});
```

Example

With data passed

```
spUtil.get('pps-list-modal', {title: c.data.editAllocation
s,
  table: 'resource_allocation',
  queryString: 'GROUPBYuser^resource_plan=' + c.data.sysId
,
  view: 'resource_portal_allocations' }).then(function(res
ponse) {
    var formModal = response;
    c.allocationListModal = response;
  });
```

spUtil - getHeaders()

Retrieve all headers to be used for API calls.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Object	All headers to be used for API calls.

spUtil - getHost()

Get complete host domain.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The complete host domain, for example `hi.servicenow.com`

spUtil - getPreference(String preference, Function callback)

Execute callback with User Preference response by passing Preference name.

Parameters

Name	Type	Description
preference	String	The name of the preference.
callback	Function	Define the callback function.

Returns

Type	Description
void	

spUtil - getURL()

Get current service portal url information.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The current service portal url.

spUtil - format(String template, Object data)

Formats a string as an alternative to string concatenation.

Use this method to build a string with variables.

Parameters

Name	Type	Description
template	String	String template with values for substitution.
data	Object	Object containing variables for substitution.

Returns

Type	Description
String	A formatted string.

Example

```
spUtil.format('An error ocurred: {error} when loading {wid
get}', {error: '404', widget: 'sp-widget'})
```

Output:

```
'An error occurred: 404 when loading sp-widget'
```

spUtil - isMobile()

Check if current client is a mobile device.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Boolean	True if the current client is a mobile device; otherwise, false

spUtil - parseAttributes(String attributes)

Brief description of the method.

Parameters

Name	Type	Description
attributes	String	String containing comma separated attributes, such as the Attributes field of a dictionary record.

Returns

Type	Description
Object	Array object containing the parsed attributes.

[Working to add example script with example output]

spUtil - recordWatch(Object $scope, String table, String filter, Function callback)

Watches for updates to a table or filter and returns the value from the callback function.

Allows a widget developer to respond to table updates in real-time. For instance, by using recordWatch(), the Simple List widget can listen for changes to its data table. If records are added, removed, or updated, the widget updates automatically.

Note: When passing the `$scope` argument into the recordWatch() function, inject `$scope` into the parameters of your client script function.

Parameters

Name	Type	Description
$scope	Object	Scope of the data object updated by the callback function.
table	String	Watched table.
filter	String	Filter for fields to watch.
callback	Function	Optional. Parameter to define the callback function.

Returns

Type	Description
Promise	Return value of the callback function.

Example

```
//A simple recordWatch function.
spUtil.recordWatch($scope, "live_profile", "sys_id=" + liv
eProfileId);

//In a widget client script
function(spUtil, $scope) {
  /* widget controller */
  var c =this;

  // Registers a listener on the incident table with the f
ilter active=true,
  // meaning that whenever something changes on that tabl
e with that filter,
  // the callback function is executed.
  // The callback function takes a single parameter 'respo
nse', which contains
  // the property 'data'. The 'data' property contains inf
ormation about the changed record.
  spUtil.recordWatch($scope, "incident", "active=true", fu
nction(response) {

    // Returns the data inserted or updated on the table
    console.log(response.data);

    });
}
```

spUtil - refresh(Object $scope)

Calls the server and replaces the current **options** and **data** with the server response.

Calling `spUtil.refresh()` is similar to calling `server.refresh()`. However, when you call `spUtil.refresh()`, you can define the $scope object.

Parameters

Name	Type	Description
$scope	Object	The scope defined for the update.

Returns

Type	Description
Object	The updated options and data objects.

spUtil - scrollTo(String selector, Number time)

Sscroll to element with specified selector, over specified period of time.

Parameters

Name	Type	Description
selector	String	The selector to scroll to.
time	Number	The time, in milliseconds, taken to scroll to the specified selector.

Returns

Type	Description
void	

spUtil - setBreadCrumb()

Update the header breadcrumbs.

Parameters

Name	Type	Description
$scope	Object	The scope defined for the table.
breadcrumbs	Array	An array of conditions used to create the breadcrumb filter.

Returns

Type	Description
void	

spUtil - setPreference(String preference, String value)

Set a user preference.

Parameters

Name	Type	Description
preference	String	The preference name
value	String	The preference value

Returns

Type	Description
void	

spUtil - setSearchPage(String searchPage)

Update the search page.

Parameters

Name	Type	Description
searchPage	String	The name of the search page.

Returns

Type	Description
void	

spUtil - update(Object $scope)

Updates the data object on the server within a given scope.

This method is similar to `server.update()`, but includes a $scope parameter that defines the scope to pass over.

Parameters

Name	Type	Description
$scope	Object	The scope defined for the update.

Returns

Type	Description
Object	The updated data object.

Example

The following example includes a P1 widget that watches for changes to the state field and uses a filter to watch all active P1s and let the callback function determine whether to refresh the data. The data.changes property contains an array of any updated fields. If the state of any fields have changed, the data is updated in the widget.

```
var q = "priority=1^active=true^EQ";
spUtil.recordWatch($scope, "incident", q, function(event,
data) {
   if (data.changes.includes("state")) { // only update i
f state was updated.
      spUtil.update($scope);
   }
});
```

StopWatch - Client

Use a StopWatch object to measure the duration of operations.

The StopWatch API can be used in client-side scripts using ListV2 and ListV3 APIs.

StopWatch - StopWatch()

Creates an instance of the StopWatch class.

Uses the current time as the start time.

Parameters

Name	Type	Description
None		

StopWatch - StopWatch(Date initialDate)

Creates an instance of the StopWatch class using the specified date as the initial value.

Parameters

Name	Type	Description
initialDate	Date	The initial date for the object.

StopWatch - getTime()

Returns the number of milliseconds since the timer started.

Parameters

Name	Type	Description
None		

Returns

Type	Description
Number	The number of milliseconds since the timer started.

StopWatch - restart()

Resets the timer to the current time.

Parameters

Name	Type	Description
None		

Returns

Type	Description
void	

StopWatch - toString()

The elapsed time as HH:MM:SS.SSS.

Parameters

Name	Type	Description
None		

Returns

Type	Description
String	The elapsed time formatted as HH:MM:SS.SSS.