Contents

API Object: OWF.Preferences.deleteUserPreference(cfg)

Definition:

Use the `/deleteUserPreference` API to remove a user preference based on its namespace and name. The preference could be user metadata such as name, email, etc.

How it works:

A system makes a call to another system that asks to reference data instead of return it. To make this call, one of the following REST Calls will request user data from the user profile URI [IS THIS RIGHT?]

Request URL:

https://localhost:8443/marketplace/api/prefs/preference/com.company.widget/First%20President

Request Method:

POST, Form_Method:Delete

Requirements:

The call must include namespace and name

- If namespace or name are null, it will change the null to 'undefined' and search for a namespace and/or name that is called 'undefined'.
- If name is "" (empty but not null). It will not throw an error but will also not delete anything.
- If namespace is "" (empty but not null) it will throw an error since the url will be "prefs/preference//name" and that is not a valid url.

Response:

If the system could not find a matching preference and there was not an error it returns:

`{"success":true, "preference":null}`

If a preference was deleted the system returns:
`{"id":7,"namespace":"com.company.widget","path":"First President","value":"foo val","user":{"userId":"te`

How to use it:

Use it to delete user preferences.

Example

The following is an example of a call to delete user preference:

```
function onSuccess(pref){
    alert(pref.value);
}

function onFailure(error,status){
    alert('Error ' + error);
    alert(status);
}


var cfg = {
    namespace: 'com.company.widget',
    name: 'First President',
    onSuccess: onSuccess,
    onFailure: onFailure
};


OWF.Preferences.deleteUserPreference(cfg);
```

Resource Information

Possible Errors

API Object: OWF.Preferences.doesUserPreferenceExist (cfg)

Definition:

Use the `/doesUserPreferenceExist` API to query if they system includes a specific user preference. It will search based on preferences' namespaces and names.

How it works:

A system makes a call to another system that asks to reference data instead of return it. To make this call, one of the following REST Calls will request user data from the user profile URI [IS THIS RIGHT?]

Request URL:

https://localhost:8443/owf/prefs/hasPreference/com.company.widget/First%20President?version=7.15.0-v1&dojo.preventCache=1433169029635

Request Method:

GET, Form_Method:none

Requirements:

The call must include namespace and name

- If namespace or name are null, it will change the null to 'undefined' and search for a namespace and/or name that is called 'undefined'.
- If name or namespace use an empty string that is not null, the query returns the HTML 404 error "The requested resource is not available."

Response:

If there is a preference matching the namespace and name it will return:

`{"preferenceExist":true,"statusCode":200}`

If there is not a matching preference it will return:

`{"preferenceExist":false,"statusCode":200}`

How to use it:

Use it to find user preferences.

Example

The following is an example of a call to delete user preference:

```
function onSuccess(pref) {alert(pref.value);}
function onFailure(error,status) {
    alert('Error ' + error);
    alert(status);
}
var cfg = {
    namespace: 'com.company.widget',
    name: 'First President',
    onSuccess: onSuccess,
    onFailure: onFailure
};

OWF.Preferences.doesUserPreferenceExist(cfg);
```

Resource Information

Possible Errors

API Object: OWF.Preferences.findWidget(cfg)

Definition:

Use the `/findWidget` API to return a Widget data for every widget in OWF.

How it works:

A system makes a call to OWF asking for metadata about every widget. To make this call, one of the following REST Calls will request widget data including the widget's associations with users and groups.

Request URL:

https://localhost:8443/marketplace/prefs/widget/listUserAndGroupWidgets

Request Method:

POST
Request Form Data
_method: GET

```
widgetName: 'widget's name'
universalName: 'widget's universal name'
widgetVersion: 'widget's version'
widgetGuid: 'widget's guid'
```

Requirements:

None

Optional: widgetName, universalName, widgetVersion, and widgetGuid (Must be inside searchParams object, see example below)

Response:

Returns a list of all widgets matching the given search parameters:

```
[
  {
    "id": "eb5435cf-4021-4f2a-ba69-dde451d12551",
    "namespace": "widget",
    "value": {
      "universalName": null,
      "namespace": "Channel Shouter",
      "description": null,
      "url": "examples/walkthrough/widgets/ChannelShouter.gsp",
      "headerIcon": "themes/common/images/widget-icons/ChannelShouter.png",
      "image": "themes/common/images/widget-icons/ChannelShouter.png",
      "smallIconUrl": "themes/common/images/widget-icons/ChannelShouter.png",
      "largeIconUrl": "themes/common/images/widget-icons/ChannelShouter.png",
      "width": 295,
      "height": 250,
      "x": 0,
      "y": 0,
      "minimized": false,
      "maximized": false,
      "widgetVersion": "1.0",
      "totalUsers": 4,
      "totalGroups": 1,
      "tags": [

      ],
      "singleton": false,
      "visible": true,
      "background": false,
      "descriptorUrl": null,
      "definitionVisible": true,
      "directRequired": [

      ],
      "allRequired": [

      ],
      "intents": {
        "send": [

        ],
        "receive": [

        ]
      },
      "widgetTypes": [
        {
          "id": 1,
          "name": "standard",
          "displayName": "standard"
        }
      ]
    },
    "path": "eb5435cf-4021-4f2a-ba69-dde451d12551"
  },
  ...more widgets...
]
```

How to use it:

Use it to compile a comprehensive list of widgets in OWF or a list of widgets with specific metadata.

Example

The following is an example of a call to find widgets:

```
function onSuccess(pref) {alert(pref.value)}
function onFailure(error,status) {
    alert('Error ' + error)
    alert(status)
}
var cfg = {
    searchParams:{
        widgetName: 'Channel Shouter'
    }
}
OWF.Preferences.findWidgets(cfg);
```

Resource Information

Possible Errors

API Object: OWF.Preferences.getUserPreference(cfg)

Definition:

Use the `/getUserPreference` API to return a user preference based on its namespace and name. The preference could be user metadata such as name, email, etc.

How it works:

A system makes a call to another system that asks to reference data instead of return it. To make this call, one of the following REST Calls will request user data from the user profile URI [IS THIS RIGHT?]

Request URL:

https://localhost:8443/owf/prefs/preference/com.company.widget/First%20President?version=7.15.0-v1&dojo.preventCache=1433176043577

Request Method:

GET, Form_Method:none

Requirements:

The call must include namespace. Name is optional.

- If name is a non-null empty string, it will return all preferences with a matching namespace.
- If namespace is a non-null empty string, it will throw an HTML 404 error, “The requested resource is not available.”
- If name or namespace are null it will fill them in with “undefined” and perform the request using ‘undefined’ as the name or namespace.

Response:

When you provide the namespace and name and the system finds a matching preference, it will return the preference information:

```
{"id":157,"namespace":"com.company.widget","path":"First President","value":"fooval","user":{"userId":"
```

How to use it:

This API provides additional metadata about a preference.

Example

The following is an example of a call to get a user preference:

```
{
  "success": true,
  "results": 3,
  "rows": [
    {
      "id": 144,
      "namespace": "com.company.widget",
      "path": "First President",
      "value": "foovalue",
      "user": {
        "userId": "testUser1"
      }
    },
    {
      "id": 146,
      "namespace": "com.company.widget",
      "path": "Second President",
      "value": "foovalue",
      "user": {
        "userId": "testUser2"
      }
    },
    {
      "id": 157,
      "namespace": "com.company.widget",
      "path": "First Vice President",
      "value": "fooval",
      "user": {
        "userId": "testAdmin1"
      }
    }
  ]
}
```

If the system does not find a matching preference, it will return:

```
{"success":true,"preference":null}
```

Example:

```
function onSuccess(pref) {alert(pref.value);}
function onFailure(error,status) {
    alert('Error ' + error);
    alert(status);
}
var cfg = {
    namespace: 'com.company.widget',
    name: 'First President',
    onSuccess: onSuccess,
    onFailure: onFailure
};

OWF.Preferences.getUserPreference(cfg);
```

Resource Information

Possible Errors

API Object: OWF.Preferences.getWidget(cfg)

Definition:

Use the `/getWidget` API to return a Widget based on its widget ID.

How it works:

A system makes a call to OWF asking if a widget exists. To make this call, one of the following REST Calls will request a widget ID.

Request URL:

https://localhost:8443/owf/prefs/widget/WIDGETID

Request Method:

GET, Form_Method:none

Requirements:

widgetID

Optional: none

Response:

OWF returns the matching widgetID, as shown below. However, if the widgetID is an empty string (“”), OWF will return a list of all widgets.

```
{
  "id": 39,
  "namespace": "widget",
  "value": {
    "originalName": "Channel Shouter",
    "universalName": null,
    "editable": true,
    "disabled": false,
    "visible": true,
    "favorite": false,
    "groupWidget": false,
    "position": 10,
    "userId": "testAdmin1",
    "userRealName": "Test Admin 1",
    "namespace": "Channel Shouter",
    "description": null,
    "url": "examples/walkthrough/widgets/ChannelShouter.gsp",
    "headerIcon": "themes/common/images/widget-icons/ChannelShouter.png",
    "image": "themes/common/images/widget-icons/ChannelShouter.png",
    "smallIconUrl": "themes/common/images/widget-icons/ChannelShouter.png",
    "largeIconUrl": "themes/common/images/widget-icons/ChannelShouter.png",
    "width": 295,
    "height": 250,
    "x": 0,
    "y": 0,
    "minimized": false,
    "maximized": false,
    "widgetVersion": "1.0",
    "groups": [

    ],
    "tags": [

    ],
    "definitionVisible": true,
    "singleton": false,
    "background": false,
    "descriptorUrl": null,
    "allRequired": [

    ],
    "directRequired": [

    ],
    "intents": {
      "send": [

      ],
      "receive": [
      ]
    },
    "widgetTypes": [
      {
        "id": 1,
        "name": "standard",
        "displayName": "standard"
      }
    ]
  },
  "path": "eb5435cf-4021-4f2a-ba69-dde451d12551"
}
```

How to use it:

Use it to find specific information about a widget including its Descriptor URL (if it has one) and its universal name.

Example

The following is an example of a call to get a widget ID:

```
var onSuccess = function(obj) {
    if (obj.value) {
        alert(obj.value.namespace);
    }
};

var onFailure = function(error) {
    alert(error);
};

OWF.Preferences.getWidget({
    widgetId:'eb5435cf-4021-4f2a-ba69-dde451d12551',
    onSuccess:onSuccess,
    onFailure:onFailure
});
```

Resource Information

Possible Errors