Recast

API Functions

Last Modified on 2026-08-25

Liquit.Initialize

The *Liquit.Initialize* function allows you to connect to the Application Workspace. It determines the version of the targeted Application Workspace zone. Then the Application Workspace API translates the API calls to the correct syntax/endpoints.

Request parameters

The first parameter is the Application Workspace zone, for which you need to specify a string containing the URL to which you want to connect.

The following extra options are available for the API as the second parameter:

Name	Description	Default value
version	The version number of the API layer to use. it is automatically detected if it is not set or it is set to null. Use the following versions to support certain feature sets: • 3.0 – This enables the feature set for 3.x. • 3.10– This enables the feature set for 4.x. It works only for Application Workspace 4.0 and later, otherwise, it returns errors on subsequent API calls.	null
relogin	Executes a custom login process when a session has expired. This can be used to trigger a new Liquit.Login/Liquit.SSO call. If this option is not configured, the page is refreshed.	null
token	Specify a token previously provided by the Login/SSO functions. It will be reused for the current session. If the token is invalid or expired, relogin will be triggered on the first API call.	null

Example

```
Liquit.Initialize('https://workspace.recastsoftware.com', {
  version: '3.0',
  relogin: function () { console.log('Session has expired'); },
  token: null
});
```

If the version of the server you specify is incorrect, the API call returns an error. The error is not returned by the *Liquit.Initialize* function.

Liquit.GetSources

The *Liquit.GetSources* function allows you to retrieve the identity sources known in the Application Workspace, as well as to determine the authentication method required by them.

Request parameters

Name	Description
description	The description provided for the identity source.
displayname	The name of the identity source which is displayed to the user.
hidden	This identity source is not listed as a selectable option but can be used to log in.
id	The id of the identity source.
methods	The methods available for the identity source • Federated – if this method is available, the *Liquit.SSO* function can be used to authenticate to this identity source. • Login – If this method is available, the *Liquit.Login* function can be used to authenticate against this identity source.

Example

```
Liquit.GetSources(
   (result ) => {

      //The result parameter will contain the identity source information.
      console.log(result);

   }
);
```

Liquit.Login

The *Liquit.Login* function allows you to connect to the Application Workspace API with provided credentials. You can specify the identity source, username and password required to authenticate to the Application Workspace.

Request parameters

You need to specify at least a username and password to authenticate to the Application Workspace.

The following parameters are available for the API:

Name	Description	Default value
identitySource	This is the name of the identity source to authenticate the user with. Leave null for default identity source.	null
username	The login username used to authenticate the user.	null
password	The login password used to authenticate the user.	null
callback	Invoked after someone tries to log in. If the login is successful, a token is generated. If the login is not successful, the response will contain an object called "fault" with a description of what caused the failure.	null

Example

```
Liquit.Login({
   identitySource: 'LOCAL',
   username : 'admin',
   password: 'password',
   callback: function (token, fault) {

      // Check if the request failed.
      if (fault != null) {
         alert('Error ' + fault.code + ': ' + fault.message);
         return;
      }

      // Call Liquit.Applications.List API to retrieve all available applications for the logged-in user.

   }
});
```

Liquit.SSO

The *Liquit.SSO* function allows a user to connect to the Application Workspace API with previously authenticated credentials that are stored in the user's web browser. It also detects whether or not a user is already authenticated to an OAuth 2 provider configured in Application Workspace.
Within it, you can specify a username or identity source required to authenticate to Application Workspace and it can be either a Microsoft Entra ID (Azure AD) or an AD FS setup.
This can be useful in scenarios where multiple OAuth2 providers are configured and used by the user.

Request parameters

Name	Description	Default value
identitySource	This is the name of the identity source to authenticate the user with. Leave null for the default identity source.	null
username	The username for login. We recommend for the best experience to set this property as the actual name of the user because it can be used as a login hint or the user's mail address (UPN).	null
popUp	If true, the authentication process is started in a popup window.	false
token	The token used to authenticate the current user on the Application Workspace Server. For more information on how to set up your identity source to enable it to grant an exchange token, see How to set up your exchange token.	
callback	Invoked after someone tries to log in. If the login is successful, a token is generated. If the login is not successful, the response will contain an object called "fault" with a description of what caused the failure.	null

Example

```
Liquit.SSO({
  username: 'admin@recastsoftware.com',
  identitySource: 'Microsoft Entra ID',
  popUp: false,
  token:null //Exchange token
}, function(token, fault) {

  // Check if authentication failed.
  if (fault != null) {
    alert('Error ' + fault.code + ': ' + fault.message);
    return;
  }

  //Do Application Workspace stuff here

});
```

Liquit.Logout

> The *Liquit.Logout* function is only available for Application Workspace Server 3.1 and later.

The *Logout* function allows you to disconnect from the Application Workspace and the provided identity source. It is possible to select whether or not only the Application Workspace session should be terminated.

Request parameters

The following parameters are available for the API:

Name	Description	Default value
excludeAction	When this parameter is set to false Application Workspace will sign out the user from the Application Workspace as well as the current identity source.	true
popUp	Controls whether or not a popup will be triggered to logout the user. If the value is set to false, an iframe will be used to trigger the logout. This parameter is available only for Application Workspace Server 3.3 and later.	true
feedback	Feedback returns false when the logout action is in progress. When the logout action is completed, feedback returns true. This parameter is available only for Application Workspace Server 3.3 and later.	

Example

```
Liquit.Logout({

  // If excludeAction is set to false, the logout function will also log you out from the selected identity source. The defa
ult value is true.
  excludeAction: false,

  // If popUp is set to true, a popup window will be used to execute the logout action. If set to false, an iframe will be u
sed.
  popUp:true

}, function (feedback) {

  // The feedback parameter is only available for Application Workspace Server 3.3 and later
  if (feedback == false) {
    console.log('Logout in progress')
  }
  if (feedback == true) {
    console.log('Logout executed.')
  }

});
```

Liquit.Agent.SelectFile

The *Liquit.Agent.SelectFile* function lets you utilize the Agent to select a local file. The Agent needs to be running to use this function.
You can check if the Agent is running by using the following variable:

```
Liquit.Workspace.Agent.Enabled
```

Usage and example

By default, the Agent will return the path where the selected file is available.

```
Liquit.Agent.SelectFile(
    function (result) {
     //result.path will be the path to the selected file
    console.log(result);
});
```

Liquit.Applications.Launch

The *Liquit.Applications.Launch* function is used to launch applications from the Application Workspace API. The applications return a callback reporting on their status: if the application has been launched successfully or not, or a status update on the progress. The update on the progress comes especially in handy when running installations or distributions.

Return parameters

The status callback property can be as follows:

Name	Description	Value
success	Application has launched successfully	String

Name	Description	Value
failure	Application has failed to start. See the details property for more information about the error.	String
inprogress	Application is being installed. Check the progress property to see how far along the installation is.	String

The details property is available only when an application has failed to launch. It reports the exact reason why an application failed to launch.

The progress property is available only when an application is in progress. It reports the progress in percentage (0–100).

Example

```
// Launch application by ID with callback returning progress information.
Liquit.Applications.Launch(id, null, function (e) {

 console.log(e);

 // *** e.state ***
 // Can be one of the following values:
 //
 // * 'success': The application has been successfully launched.
 // * 'inprogress': The application is being installed. Check the progress property to see how far along the installation i
s.
 // * 'failure': The application has failed to start. See the details property for more information about the error.
 //
 // *** e.progress ***
 // Gives an estimated percentage of the installation progress.
 //
 // *** e.details ***
 // Gives additional information about the failure to launch the application.
 // This is structured with a 'code' field and a 'message' field describing the error.
 //

 // Application started
 if (e.state == 'success')
  alert('application started');

 // Failed to start application.
 else if (e.state == 'failure')
  alert('application failed to start: ' + e.details.code + ': ' + e.details.message);

});
```

Liquit.Applications.List

The *Liquit.Applications.List* function returns all applications that are available to the user. Whenever an application has been made available in the Application Workspace it will be listed in the response of this function. The Application Workspace API autodetects whether or not the Agent is available and adjusts its response according to it. For example, a local application will not be enabled when the Agent is not running.

Request parameters

Recast

Name	Description	Value
sort	Determines the order in which the API returns the applications: • 0 User-defined ordering, as determined by their position • 1 Name ascending • 2 Name descending • 3 By launch count (available for Application Workspace 3.4 and later) • 4 By last launch (available for Application Workspace 3.4 and later)	<int>
refresh	Defines if the action is a refresh.	```
search	Allows you to search by name	<string>
tags	If its value is true, tags will be included	<boolean>
teams	If its value is true, teams will be included	<boolean>
event	Define the type of event: • **None** no action is executed. • **Refresh** triggers actions defined in refresh events. • **LiquitLogon** triggers the actions that would be executed at Application Workspace logon.	<string>

Return parameters

The following properties are returned:

Name	Description	Value
autoLaunch	Whether or not auto launch is enabled for the application	<boolean>
categories	Returns an array of categories in which the application is included	<array[]>
editable	Determines whether or not the end-user is privileged to edit the package	<boolean>
enabled	Determines whether or not the application is enabled (e.g. local applications will not be enabled when no Agent is running.)	<boolean>
favorite	Will return true if the end-user added this application to their favorites	<boolean>
icon	Returns the URL of the application icon	<URL>
id	The id of the application	<guid>
name	The friendly name of the application	<string>
new	Will return true if the end-user has never started this application	<boolean>
position	The position of the application	<int>
rating	This will return the average rating of the app calculated based on the rating given by all users	<int>
removable	Whether or not the end-user is allowed to remove the application from their workspace (e.g. forced applications will return false)	<boolean>
repair	Whether or not the repair action is available for the application	<boolean>
source	Returns an object with information about the applications: • **id** the ID of the source which can be either the ID of the zone or of a team • **type** the source type, which can be either “Zone” or “Team”	<object{}>

Recast

Name	Description	Value
stage	The published stage: • Development • Test • Acceptance • Production	<string>
status	Returns the state of the application.	<string>

Example

Below you will find a small example of code to render the applications and their respective icons.

```
// Do Liquit API calls...
Liquit.Applications.List({

  sort:0

},function (fault, result) {

  // Check if the request failed.
  if (fault != null) {
    alert('Error ' + fault.code + ': ' + fault.message);
    return;
  }

  for (var i = 0; i < result.length; i++) {

    $('body').append(
      $('<div>')
        .attr({
          id: result[i].id
        })
        .addClass('app')
        .toggleClass('disabled', !result[i].enabled)
        .on('click', function (e) {

          // Get target html element.
          var target = $(e.currentTarget);

          // Verify application is disabled
          if (target.hasClass('disabled')) {
            alert('Application is disabled');
            return;
          }

          // Launch application by ID with callback returning progress information.
          Liquit.Applications.Launch(target.attr('id'), null, function (e) {

            console.log(e);

            // *** e.state ***
            // Can be one of the following values:
            //
            // * 'success': The application has been successfully launched.
            // * 'inprogress': The application is being installed. Check the progress property to see how far along the install
ation is.
            // * 'failure': The application has failed to start. See the details property for more information about the error.
            //
```

```
          //
          // *** e.progress ***
          // Gives an estimated percentage of the installation progress.
          //
          // *** e.details ***
          // Gives additional information about the failure to launch the application.
          // This is structured with a 'code' field and a 'message' field describing the error.
          //

          // Application started
          if (e.state == 'success')
            alert('application started');

          // Failed to start application.
          else if (e.state == 'failure')
            alert('application failed to start: ' + e.details.code + ': ' + e.details.message);

        });

      })
      .append(
        $('<img>')
          .addClass('icon')
          .attr({
            src: result[i].icon
          })
          .toggle(result[i].icon != null),
        $('<span>')
          .addClass('title')
          .text(result[i].name)
      )
    );

  }

});
```

Liquit.Applications.Create

The *Liquit.Applications.Create* function lets the end-user add a personal application of their own. This can be a local application or a URL the user wants to save to their workspace.
By default, the *Liquit.Application.Create* API will return the ID of the freshly created personal application. When the request fails, an error will be returned in the fault object of the response. This can be because an application with the same name already exists.

Request parameters

Based on the type, the following parameters are available:

Name	Description	Value	Required
type	This can be either **path** or **url**	`<string>`	Yes
name	The name of the personal package	`<string>`	Yes
url	The URL that the personal package should open	`<string>`	Personal URI only

Recast

Name	Description	Value	Required
path	The path to the executable that should be opened by the Agent	`<string>`	Local path only
icon	The file object used for the icon	`<file>`	No
directory	The start directory the application will use when launched.	`<string>`	Local path only
display	Defines how the window for the application should be opened: • *Normal* – Open up the application in the default settings. • *Minimized* – Start the application minimized. • *Maximized* – Start the application maximized.	`<string>`	Local path only
browser	The browser used to open the webpage. • Default • InternetExplorer • Firefox • Chrome • Opera • Edge • Safari	`<string>`	Personal URI only

Example

```
var type = 'url';
var name = 'friendlyName';
var path = 'https://www.recastsoftware.com';
var icon = $('.icon')[0].files;

Liquit.Applications.Create({
  type: type,
  name: name,
  url: path,
  icon: file
}, function (fault, result) {
  console.log(fault, result);
});
```

Liquit.Applications.Favorite

The *Liquit.Applications.Favorite* function adds an application to the favorites list of the end-user. This function requires the ID of the application you want to add to the favorites list.

Usage and example

You need to specify at least the values shown in the example below:

```
Liquit.Applications.Favorite(

  // The id of the application you want to favorite or unfavorite.
  '00000000-0000-0000-0000-000000000000',

  // True adds the application to the favorites, false removes the application from the favorites.
  true,

  // Callback which either returns an error or the result of the API call.
  function (fault, result) {
    console.log(result);
  }

);
```

Liquit.Applications.Get

The *Liquit.Applications.Get* function returns the properties of a personal application. This function is not available for packages assigned to the user via the Application Workspace interface.
By default, the *Liquit.Applications.Get* API returns information about the entity triggered by the personal app, and its corresponding settings. To acquire information on the application, you need to supply its ID and its source information.

Request parameters

The following parameters are available for the API:

Name	Description	Value
id	The ID of the application you wish to request.	`<GUID>`
source.type	This information is made available to you by the *Applications.List* function.	`<string>`
source.id	This information is made available to you by the *Applications.List* function.	`<GUID>`

Return parameters

The following properties are returned:

Name	Description	Value
browser	Defines the browser that will be used to start the application: • Default • InternetExplorer • Firefox • Chrome • Opera • Edge • Safari Only available for a **URI** application.	`<string>`
directory	The start directory the application will use when launched. Only available for a **path** application.	`<string>`

Name	Description	Value
display	Defines how the window for the application should be opened: • **Normal:** Open up the application in the default settings. • **Minimized:** Start the application minimized. • **Maximized:** Start the application maximized. Only available for a **path** application.	<string>
id	The ID of the application.	<string>
name	The display name of the personal application.	<string>
path	The path of the local application. Only available for a **path** application.	<string>
type	The type of application: • **Path:** a local application. These applications require an Agent to be running. • **Uri:** A URI that can be opened by the browser. These applications do not require an Agent to be running).	<string>
uri	The URI the package will open. Only available for a **URI** application.	<string>

Example

Below you will find a small example of code to fetch a personal application and its starting information.

```
// Do Application Workspace API calls...
Liquit.Applications.Get({
    id: id,
    source: {
        id: sourceId,
        type: 'Team'
    }
}, function(fault, result) {
    console.log({
        fault: fault,
        result: result
    });
});
```

Liquit.Applications.Move

The *Liquit.Applications.Move* function changes the position of an application. This determines the order of the applications shown if no explicit sorting options are used in the *Liquit.Applications.List* function.

Usage and example

You need to specify at least the ID of the targeted application and the target position:

```
Liquit.Applications.Move(

  // The id of the application you want to move.
  '00000000-0000-0000-0000-000000000000',

  // The target position.
  0,

  // Callback that will return a fault object when something goes wrong.
  function (fault, result) {
    console.log(result);
  }

);
```

Liquit.Applications.Remove

The *Liquit.Applications.Remove* function removes an application from the Application Workspace.

Example

You need to specify at least the ID of the application you want to remove as shown in the example below:

```
Liquit.Applications.Remove(

  // The id of the application you want to remove from the workspace.
  '00000000-0000-0000-0000-000000000000',

  // The callback function that provides feedback.
  function (fault, result) {
    console.log(result);
  }

);
```

Liquit.Applications.Update

The *Liquit.Applications.Update* function updates a personal application.
By default, the *Liquit.Application.Update* API returns the ID of the freshly created personal application. When the request fails, an error is returned in the fault object of the response. This can can happen for example because an application with the same name already exists.

Request parameters

Based on the type of application, the following options are available:

Name	Description	Value	Required
type	This can be either **path** or **url**.	`<string>`	Yes
name	The name of the personal package.	`<string>`	Yes
url	The URL that the personal package should open.	`<string>`	Personal URI only
path	The path to the executable that should be opened by the Agent.	`<string>`	Local path only

Name	Description	Value	Required
icon	The file object used for the icon.	<file>	No
directory	The start directory the application will use when launched.	<string>	Local path only
display	Defines how the window for the application should be opened: • **Normal:** Open up the application according to the default settings. • **Minimized:** Start the application minimized. • **Maximized:** Start the application maximized.	<string>	Local path only
browser	The browser used to open the webpage: • Default • InternetExplorer • Firefox • Chrome • Opera • Edge • Safari	<string>	Personal URI only

Example

```
var type = 'url';
var name = 'friendlyName';
var path = 'https://www.recastsoftware.com';
var icon = $('.icon')[0].files;

Liquit.Applications.Update('xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' , {
  type: type,
  name: name,
  url: path,
  icon: file
}, function (fault, result) {
  console.log(fault, result);
});
```

Liquit.Catalog.List

The *Liquit.Catalog.List* function returns all applications available in the user's catalog. Whenever an application is made available in the **Catalog**, it is listed in the response of this function.

Request parameters

The following parameters are available for the API:

Name	Description	Value
sort	Determines the order in which the API returns the applications: • **0** User-defined ordering, as determined by their position • **1** Name ascending • **2** Name descending • **3** By launch count (available for Application Workspace 3.4 and later) • **4** By last launch (available for Application Workspace 3.4 and later)	<int>

Name	Description	Value
search	Allows you to search by name	<string>
reviews	If true, reviews will be included in the response (available for Application Workspace 3.6 and later)	<boolean>
media	If true, media will be included in the response (available for Application Workspace 3.6 and later)	<boolean>
faq	If true, the frequently asked questions will be included in the response (available for Application Workspace 3.6 and later)	<boolean

Return parameters

Name	Description	Value
active	Determines whether or not the application is active. If this is true, the application was already successfully requested.	<boolean>
categories	Returns an array containing all categories associated with the application.	<array[]>
icon	Returns the URL of the application icon.	<string>
id	The GUID of the application.	<guid>
name	The friendly name of the application.	<string>
rating	The global rating of the application.	<int>
requestable	Indicates if the application is requestable by the user	<boolean>
status	The status of the application: • *active* – The application has been requested and approved • *available* – The application can be requested and no approval workflow is configured, meaning it becomes available in the workspace immediately • *pending* – The application has been requested, however, an approval is still needed	<string>

Example

```
Liquit.Catalog.List({
    media: true,
    license: true,
    reviews: true,
    faq: true
        }, function (fault, result) {

        // Check if the request failed.
        if (fault != null) {
            alert('Error ' + fault.code + ': ' + fault.message);
            return;
     }

     // The result object will contain an array of all catalog items.
        console.log(result);
});
```

Liquit.Catalog.Get

Recast

The *Liquit.Catalog.Get* function returns all information on an application in the user catalog. Whenever an application has been made available in the **Catalog** it will be available in the response of this function.

Request parameters

The following parameters are available for the API:

Name	Description	Value
id	The GUID needed to request the application	<GUID>
reviews	If true, reviews will be included in the response (available for Application Workspace 3.6 and later)	<boolean>
media	If true, media will be included in the response (available for Application Workspace 3.6 and later)	<boolean>
faq	If true, the frequently asked questions will be included in the response (available for Application Workspace 3.6 and later)	<boolean

Return parameters

The following properties are returned:

Name	Description	Value
active	Determines whether or not the application is active. If this is true, the application was already successfully requested.	<boolean>
categories	Returns an array containing all categories associated with the application.	<array[]>
icon	Returns the URL of the application icon.	<string>
id	The GUID of the application.	<guid>
name	The friendly name of the application.	<string>
rating	The global rating of the application.	<int>
status	The status of the application: • *active* – The application has been requested and approved • *available* – The application can be requested and no approval workflow is configured, meaning it becomes available in the workspace immediately • *pending* – The application has been requested, however, an approval is still needed	<string>

Example

```
var id = '00000000-0000-0000-0000-000000000000';
// Catalog get
Liquit.Catalog.Get({
 id:id,
 media: true,
 license: true,
 reviews: true,
 faq: true
}, function (fault, result) {
 console.log(result)
});
```

Recast

Liquit.Categories.List

The **Liquit.Categories.List** function returns all categories available to the user. Categories can be assigned to applications and help organize them. To get insight into the assigned categories, see Liquit.Applications.List or Liquit.Applications.Get.

Return parameters

The API call returns an array with all the categories assigned to the user. Each category will have the following properties.

Name	Description	Value
id	The unique identifier of the category.	<guid>
color	The color associated with the category. The value is a hex value for the color.	<string>
description	The description of the category.	<string>
name	The friendly name of the category.	<string>

Example

```
Liquit.Categories.List(function (result) {

    // The result object will contain an array of all categories.
      console.log(result);
});
```

Liquit.Catalog.Request

The *Liquit.Catalog.Request* function is used to request applications from catalog. All applicable applications return a callback reporting on the status of the application, this can be either the application has been requested successfully or indicate approval from an approver is equired.

Return parameters

The status callback property can be as follows:

Name	Description	Value
Null	The application has been requested successfully and is assigned to the users' workspace	Null
Pending	The application needs approval before being assigned to the users workspace.	String

```
Liquit.Catalog.Request(id, function (e) {
// e = null
// the requested application has been approved and the application becomes available in the workspace right away

// e = “Pending”

// The requested application needs approval before becoming available in the workspace.

});
```

Recast

Liquit.Settings.List

The *Liquit.Settings.List* function returns all user and portal settings available. Some properties are read-only; for more details, see the list with all properties.

Usage

Every setting has at least four different properties:

- *id* – The identifier of the setting (e.g. "portal.animations" identifies whether or not animations are enabled)
- *force* – Reflects whether or not an administrator has locked a user setting in Application Workspace >**Manage** > **User Settings**.
- *inherited* – This property reflects whether or not the value is inherited or if it is modified
- *value* – Returns the value of the setting.

Example

```
Liquit.Settings.List(function (fault, result) {

  // Check if the request failed.
  if (fault != null) {
    alert('Error ' + fault.code + ': ' + fault.message);
    return;
  }

  // The result object contains an array with all the settings.
  console.log(result)

  // Build a table with all the available settings and their values.
  var table = $('<table>')
  .append(
    $('<tr>').append(
      $('<td>')
      .text('Setting id'),
      $('<td>')
      .text('Value'),
      $('<td>')
      .text('Force'),
      $('<td>')
      .text('Inherited')
    )
  );

  // Loop through result.
  for (var i in result)
    table.append(
      $('<tr>').append(
        $('<td>').text(result[i].id),
        $('<td>').text(result[i].value),
        $('<td>').text(result[i].force),
        $('<td>').text(result[i].inherited)
      )
    )

  table.appendTo($('body'));

});
```

Recast

Liquit.Settings.Update

The *Liquit.Settings.Update* function enables you to update a setting to a new value. Some properties are read-only; for more details, see the list with all properties.

Usage

To update a setting you need to at least specify the ID of the setting and the value. It is possible to update more than one setting at a time.

Example

```
Liquit.Settings.Update(

 // The array with the settings you want to adjust.
 [
   {

    // The id of the setting you want to adjust.
    id: 'portal.background.color',

    // The value you want to assign to the property.
    value: '000'

   },
   {
    id: 'portal.text.shadow',
    value:false
   }
 ],

 // Callback.
 function (fault, result) {

  // If an error occurs, the fault object will return an error and an error message.
  console.log(fault);

  // Returns the result of the API call when connected to Application Workspace 3.0 and later. Application Workspace
2.x APIs will return null
  console.log(result);

 }

);
```

Liquit.Contexts.List

> This API function is available for Application Workspace 3.5 and later.

The *Liquit.Contexts.List* function returns all contexts that are available to the user.

Recast

Return parameters

The following properties are available:

Name	Description	Value
id	The GUID of the context	<guid>
name	The friendly name of the context	<string>
primary	Only one context will be primary; this is used for filtering in the Application Workspace.	<boolean>
priority	This number indicates which context is more important, the lower the number the higher the priority (e.g. 0 is more important than 2).	<number>

Example

```
Liquit.Contexts.List(function (fault, result) {

      // Check if the request failed.
      if (fault != null) {
         alert('Error ' + fault.code + ': ' + fault.message);
         return;
   }

   // The result object will contain an array of all contexts
      console.log(result);
});
```

Liquit.Tags.List

The **Liquit.Tags.List** function returns all tags available to the user. Tags can be assigned to applications and help organize applications. To get insight into the assigned tags, see Liquit.Applications.List or Liquit.Applications.Get.

Return parameters

The API call returns an array with all the tags assigned to the user. These are available because they are assigned to applications that in turn are assigned to the user. Each tag will have the following properties.

Name	Description	Value
id	The unique identifier of the tag.	<guid>
color	The color associated with the tag, the value is a hex value for the color.	<string>
description	The description of the tag.	<string>
name	The friendly name of the tag.	<string>

Example

```
Liquit.Tags.List(function (result) {

   // The result object will contain an array of all tags.
      console.log(result);
});
```

Recast