

# i18n

## Method Summary

`String[]` `#getAvailableTimeZoneIds()`  
Get the list of available time zones.

`String[]` `#getCountries()`  
Gets the list of countries available for localization

`String` `#getCurrentCountry()`  
Gets the current country; based on the current locale settings in the Servoy Client Locale preferences.

`String` `#getCurrentLanguage()`  
Gets the current language; based on the current locale settings in the Servoy Client Locale preferences.

`String` `#getCurrentTimeZone()`  
Gets the current time zone of the client; based on the current locale settings in the Servoy Client Locale preferences.

`String` `#getDefaultDateFormat()`  
Gets the current default date format; based on the current locale settings in the Servoy Client Locale preferences.

`String` `#getDefaultNumberFormat()`  
Gets the current default number format; based on the current locale settings in the Servoy Client Locale preferences.

`String` `#getI18NMessage(i18nKey)`  
Gets the real message (for the clients locale) for a specified message key.

`String` `#getI18NMessage(i18nKey, dynamicValues)`  
Gets the real message (for the clients locale) for a specified message key.

`JSDat` `#getLanguages()`  
`aSet` Returns a dataset with rows that contains a language key (en) and the displayname (English) column.

`JSDat` `#getSystemMessages()`  
`aSet` Returns a dataset with rows that contains 3 columns: 'key' (i18n key), 'reference' (reference text for that key) and 'locale ([CURRENT\_LOCALE])' (where [CURRENT\_LOCALE] is the current language) - with the system messages of servoy.

`Number` `#getTimeZoneOffset(timeZoneId)`  
`er` Returns the offset (in milliseconds) of this time zone from UTC for the current date or at the specified date.

`Number` `#getTimeZoneOffset(timeZoneId, date)`  
`er` Returns the offset (in milliseconds) of this time zone from UTC for the current date or at the specified date.

`String[]` `#getTimeZones()`  
Returns an array of known timezones.

`void` `#setI18NMessage(i18nKey, value)`  
Sets the value of i18n key for client scope, if value null the setting is removed.

`void` `#setI18NMessagesFilter(columnName, value)`  
Call this if you want to add a filter for a column (created by you) in the i18n table.

`void` `#setI18NMessagesFilter(columnName, value)`  
Call this if you want to add a filter for a column (created by you) in the i18n table.

`void` `#setLocale(language, country)`  
Set/Overwrite the locale for this client.

`void` `#setTimeZone()`  
Sets the current time zone of the client.

## Method Details

`getAvailableTimeZoneIds`

`String[]` **`getAvailableTimeZoneIds()`**

Get the list of available time zones.

### Returns

`String[]`

### Sample

```
//Get the list of available time zones
var timezones = i18n.getAvailableTimeZoneIds();
```

`getCountries`

`String[]` **`getCountries()`**

Gets the list of countries available for localization

### Returns

`String[]` – a String array containing the available countries.

### Sample

```
i18n.getCountries()
```

getCurrentCountry

**String** **getCurrentCountry()**

Gets the current country; based on the current locale settings in the Servoy Client Locale preferences.

NOTE: For more information on i18n, see the chapter on Internationalization (i18n) in the Servoy Developer User's Guide, beginning with the Introduction to i18n

### Returns

**String** – a String representing the current country.

### Sample

```
var currCountry = i18n.getCurrentCountry();
```

getCurrentLanguage

**String** **getCurrentLanguage()**

Gets the current language; based on the current locale settings in the Servoy Client Locale preferences.

NOTE: For more information on i18n, see the chapter on Internationalization (i18n) in the Servoy Developer User's Guide, beginning with the Introduction to i18n

### Returns

**String** – a String representing the current language.

### Sample

```
var currLang = i18n.getCurrentLanguage();
```

getCurrentTimeZone

**String** **getCurrentTimeZone()**

Gets the current time zone of the client; based on the current locale settings in the Servoy Client Locale preferences. For Servoy Web Clients the time zone is given by the browser (if it is possible to obtain it).

### Returns

**String** – a String representing the current time zone.

### Sample

```
var currTimeZone = i18n.getCurrentTimeZone();
```

getDefaultDateFormat

**String** **getDefaultDateFormat()**

Gets the current default date format; based on the current locale settings in the Servoy Client Locale preferences.

### Returns

**String** – a String representing the default date format.

### Sample

```
var defaultDateFormat = i18n.getDefaultDateFormat();
```

getDefaultNumberFormat

**String** **getDefaultNumberFormat()**

Gets the current default number format; based on the current locale settings in the Servoy Client Locale preferences.

### Returns

**String** – a String representing the default number format.

### Sample

```
var defaultNumberFormat = i18n.getDefaultNumberFormat();
```

getI18NMessage

**String** **getI18NMessage(i18nKey)**

Gets the real message (for the clients locale) for a specified message key.

### Parameters

[{String}](#) i18nKey – The message key

### Returns

[String](#) – a String that is the message for the message key.

### Sample

```
// returns 'Welcome my_name in my solution'
// if the key 'mykey.username.text' is 'Welcome {0} in my solution'
i18n.getI18NMessage('mykey.username.text',new Array('my_name'))
```

### getI18NMessage

[String](#) **getI18NMessage**(i18nKey, dynamicValues)

Gets the real message (for the clients locale) for a specified message key.

You can use parameter substitution by using {n}, where n is a index number of the value thats in the arguments array.

### Parameters

[{String}](#) i18nKey – The message key

[{Object\[\]}](#) dynamicValues – Arguments array when using parameter substitution.

### Returns

[String](#) – a String that is the message for the message key.

### Sample

```
// returns 'Welcome my_name in my solution'
// if the key 'mykey.username.text' is 'Welcome {0} in my solution'
i18n.getI18NMessage('mykey.username.text',new Array('my_name'))
```

### getLanguages

[JSDataset](#) **getLanguages**()

Returns a dataset with rows that contains a language key (en) and the displayname (English) column.

See <http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt> for a list that could be returned.

### Returns

[JSDataset](#) – a JSDataset with all the languages.

### Sample

```
var set = i18n.getLanguages();
for(var i=1;i<=set.getMaxRowIndex();i++)
{
    application.output(set.getValue(i, 1) + " " + set.getValue(i, 2));
}
```

### getSystemMessages

[JSDataset](#) **getSystemMessages**()

Returns a dataset with rows that contains 3 columns: 'key' (i18n key), 'reference' (reference text for that key) and 'locale' ([CURRENT\_LOCALE]) (where [CURRENT\_LOCALE] is the current language) - with the system messages of servoy.

This means all servoy messages, with all available translations.

### Returns

[JSDataset](#) – a JSDataset with all the system messages.

### Sample

```
var set = i18n.getSystemMessages();
for(var i=1;i<=set.getMaxRowIndex();i++)
{
    application.output(set.getValue(i, 1) + " " + set.getValue(i, 2)+ " " + set.getValue(i, 3));
}
```

### getTimeZoneOffset

[Number](#) **getTimeZoneOffset**(timeZoneld)

Returns the offset (in milliseconds) of this time zone from UTC for the current date or at the specified date.

### Parameters

[{String}](#) timeZoneld – The time zone to get the offset for.

### Returns

[Number](#) – an int representing the time zone's offset from UTC.

### Sample

```
var timeZoneOffset = i18n.getTimeZoneOffset('America/Los_Angeles');
```

#### getTimeZoneOffset

**Number** **getTimeZoneOffset**(timeZoneId, date)

Returns the offset (in milliseconds) of this time zone from UTC for the current date or at the specified date.

#### Parameters

**{String}** timeZoneId – The time zone to get the offset for.

**{Date}** date – The date in the time zone (default current date). Needed in case daylight saving time/GMT offset changes are used in the time zone.

#### Returns

**Number** – an int representing the time zone's offset from UTC.

### Sample

```
var timeZoneOffset = i18n.getTimeZoneOffset('America/Los_Angeles');
```

#### getTimeZones

**String[]** **getTimeZones**()

Returns an array of known timezones.

#### Returns

**String[]** – an Array with all the timezones.

### Sample

```
var timeZones = i18n.getTimeZones();
```

#### setI18NMessage

void **setI18NMessage**(i18nKey, value)

Sets the value of i18n key for client scope, if value null the setting is removed.

All forms not yet loaded will change (execute this in solution startup or first form)

#### Parameters

**{String}** i18nKey – The message key

**{String}** value – The value for the message key.

#### Returns

void

### Sample

```
//sets the value of i18n key for client scope; if value null the setting is removed
//Warning: already created form elements with i18n text lookup will not change,
//so call this method in the solution startup method or in methods from first form
//this method saves message for current locale, so if locale is changed with setLocale, all messages set from
scripting will be lost

i18n.setI18NMessage('mykey.username.text', 'my_name')
```

#### setI18NMessagesFilter

void **setI18NMessagesFilter**(columnName, value)

Call this if you want to add a filter for a column (created by you) in the i18n table.

So that you can have multiple default values and multiple values per locale for one key.

#### Parameters

**{String}** columnName – The column name that is the filter column in the i18n table.

**{String}** value – The filter value.

#### Returns

void

## Sample

```
// Puts i18n in filter mode - this allows you to have multiple default/per locale
// values for one key and to use one of them based on the filter parameters.
// Let's say you added a new column "message_variant" to your i18n table.
// Now you can have keys that will translate to a language differently depending on the used variant
// For example you have 2 rows in you table for key X, language EN, different values and different
"message_variant" (let's say 1 and 2)
// If you want the solution to use the first variant, you will have to call:
i18n.setI18NMessagesFilter('message_variant', '1')

// Using filter fall back : if there is no key for 'message_variant' with value '1', it will
// search for value '2'
var filterValue = new Array();
filterValue[0] = '1';
filterValue[1] = '2';
i18n.setI18NMessagesFilter('message_variant', filterValue)

// ATTENTION: if you use setI18NMessagesFilter(...) it is not recommended to use the i18n Dialog (especially
before the filter is applied through JS).
```

### setI18NMessagesFilter

void **setI18NMessagesFilter**(columnName, value)

Call this if you want to add a filter for a column (created by you) in the i18n table.

So that you can have multiple default values and multiple values per locale for one key.

You can use filter fall back by setting the filter value as an array of strings.

#### Parameters

{String} columnName – The column name that is the filter column in the i18n table.

{String[]} value – The filter value.

#### Returns

void

## Sample

```
// Puts i18n in filter mode - this allows you to have multiple default/per locale
// values for one key and to use one of them based on the filter parameters.
// Let's say you added a new column "message_variant" to your i18n table.
// Now you can have keys that will translate to a language differently depending on the used variant
// For example you have 2 rows in you table for key X, language EN, different values and different
"message_variant" (let's say 1 and 2)
// If you want the solution to use the first variant, you will have to call:
i18n.setI18NMessagesFilter('message_variant', '1')

// Using filter fall back : if there is no key for 'message_variant' with value '1', it will
// search for value '2'
var filterValue = new Array();
filterValue[0] = '1';
filterValue[1] = '2';
i18n.setI18NMessagesFilter('message_variant', filterValue)

// ATTENTION: if you use setI18NMessagesFilter(...) it is not recommended to use the i18n Dialog (especially
before the filter is applied through JS).
```

### setLocale

void **setLocale**(language, country)

Set/Overwrite the locale for this client.

All forms not yet loaded will change (execute this in solution startup or first form).

The language must be a lowercase 2 letter code defined by ISO-649.

see <http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt>

The country must be an upper case 2 letter code defined by IS-3166

see [http://www.chemie.fu-berlin.de/diverse/doc/ISO\\_3166.html](http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html)

NOTE: For more information on i18n, see the chapter on Internationalization (i18n) in the Servoy Developer User's Guide, beginning with the Introduction to i18n

#### Parameters

{String} language – The lowercase 2 letter code

{String} country – The upper case 2 letter code.

**Returns**

void

**Sample**

```
//Warning: already created form elements with i18n text lookup will not change,  
//so call this method in the solution startup method or in methods from first form  
  
i18n.setLocale('en','US');
```

**setTimeZone**

void **setTimeZone()**

Sets the current time zone of the client.

The parameter should be a string having the format which can be retrieved via `i18n.getAvailableTimeZones` or can be seen in the SmartClient Edit -> Preferences -> Locale at the "Default Timezone" combobox.

For instance the time zone for Netherlands is set using the ID "Europe/Amsterdam".

**Returns**

void

**Sample**

```
// This will set the default time zone to Central European Time  
i18n.setTimeZone("Europe/Amsterdam");
```