$form\_state is now a classed object

Primary tabs

* [View(active tab)](https://www.drupal.org/node/2310411)
* [View history](https://www.drupal.org/node/2310411/revisions)

Posted by [tim.plunkett](https://www.drupal.org/u/timplunkett) on 31 July 2014

**Change record status:**

Published ([View all published change records](https://www.drupal.org/list-changes))

**Project:**

[Drupal core](https://www.drupal.org/project/drupal)

**Introduced in branch:**

8.x

**Issues:**

[#2225353: Convert $form\_state to an object and provide methods like setError()](https://www.drupal.org/node/2225353)

[#2308821: Replace FormErrorInterface with $form\_state->setErrorByName() and $form\_state->setError()](https://www.drupal.org/node/2308821)

[#2313479: Add FormState::setResponse()](https://www.drupal.org/node/2313479)

[#2313823: Use FormStateInterface for all typehints](https://www.drupal.org/node/2313823)

[#2316533: Add getValue/setValue/hasValue and isValueEmpty to FormState](https://www.drupal.org/node/2316533)

[#2318087: Replace $form\_state['input'] with FormState::getUserInput()](https://www.drupal.org/node/2318087)

[#2332389: Finish adding methods to FormStateInterface](https://www.drupal.org/node/2332389)

[#2335659: Remove FormState ArrayAccess usage from core](https://www.drupal.org/node/2335659)

**Description:**

Since its introduction in [#138706: FormAPI 3: Ready to rock](https://www.drupal.org/node/138706), the form state has been an array. The form state tracks the "current state" of the form as it is built, processed, validated, and submitted.

In Drupal 8, the newly added [FormStateInterface](https://api.drupal.org/api/drupal/core%21lib%21Drupal%21Core%21Form%21FormStateInterface.php/interface/FormStateInterface/8) is used instead of an array for $form\_state.

The following code snippets reference the FormBuilder. See [drupal\_\*\_form() methods are replaced by a form builder service](https://www.drupal.org/node/2121003) for more information.

Before:

$form\_state['rebuild'] = TRUE;

$form\_builder->buildForm('Drupal\my\_module\Form\MyForm', $form\_state);

After:

use Drupal\Core\Form\FormState;

$form\_state = new FormState();

$form\_state->setRebuild();

$form\_builder->buildForm('Drupal\my\_module\Form\MyForm', $form\_state);

In order to do things like setting errors, you no longer need the whole form builder:

Before:

form\_set\_error($element, $form\_state, $message);

form\_set\_error('element\_name', $form\_state, $message);

After:

$form\_state->setError($element, $message);

$form\_state->setErrorByName($element, $message);

In addition to the functions that were replaced by methods, all array-based usage of $form\_state is now replaced by methods:  
Before:

$complete\_form = &$form\_state['complete\_form'];

$form\_state['complete\_form'] = &$complete\_form;

After:

$complete\_form = &$form\_state->getCompleteForm();

$form\_state->setCompleteForm($complete\_form);

Before:

$response = $form\_state['response'];

$form\_state['response'] = $response;

After:

$response = $form\_state->getResponse();

$form\_state->setResponse($response);

Before:

$form\_state['redirect'] = new Url('route\_name', array('route' => 'parameters'));

$form\_state['redirect'] = array('route\_name', array('route' => 'parameters'));

$redirect = $form\_state['redirect'];

$form\_state['no\_redirect'] = TRUE;

if (!empty($form\_state['no\_redirect'])) {

}

After:

$form\_state->setRedirectUrl(new Url('route\_name', ['route' => 'parameters']));

$form\_state->setRedirect('route\_name', ['route' => 'parameters']);

$redirect = $form\_state->getRedirect();

$form\_state->disableRedirect();

if ($form\_state->isRedirectDisabled()) {

}

Before:

$storage = &$form\_state['storage'];

$form\_state['storage'] = &$storage;

After:

$storage = &$form\_state->getStorage();

$form\_state->setStorage($storage);

Before:

$temporary = $form\_state['temporary'];

$form\_state['temporary'] = $temporary;

After:

$temporary = $form\_state->getTemporary();

$form\_state->setTemporary($temporary);

Before:

$arbitrary\_value = $form\_state['arbitrary\_key'];

$form\_state['arbitrary\_key'] = $arbitrary\_value;

$nested\_arbitrary\_value = $form\_state['nested']['arbitrary\_key'];

$form\_state['nested']['arbitrary\_key'] = $nested\_arbitrary\_value;

After:

$arbitrary\_value = $form\_state->get('arbitrary\_key');

$form\_state->set('arbitrary\_key', $arbitrary\_value);

$nested\_arbitrary\_value = $form\_state->get(['nested', 'arbitrary\_key']);

$form\_state->set(['nested', 'arbitrary\_key'], $nested\_arbitrary\_value);

Before:

$build\_info = $form\_state['build\_info'];

$form\_state['build\_info'] = $build\_info;

$form\_state['build\_info']['args'] = $args;

$args = $form\_state['build\_info']['args'];

After:

$build\_info = $form\_state->getBuildInfo();

$form\_state->setBuildInfo($build\_info);

$form\_state->addBuildInfo('args', $args);

$args = $form\_state->getBuildInfo()['args'];

Before:

$input = &$form\_state['input'];

$form\_state['input'] = $input;

After:

$input = &$form\_state->getUserInput();

$form\_state->setUserInput($input);

Before:

$foo = $form\_state['values']['foo']; // This will return an array. print\_r($foo)

$foobar = $form\_state['values']['foo']['bar'];

$values = $form\_state['values'];

$form\_state['values']['foo'] = 'bar';

$form\_state['values']['bar']['baz'] = 'foo';

if (isset($form\_state['values']['foo'])) {

}

if (isset($form\_state['values']['foo']['bar'])) {

}

if (empty($form\_state['values']['foo'])) {

}

if (empty($form\_state['values']['foo']['bar'])) {

}

After:

$foo = $form\_state->getValue('foo');

$foobar = $form\_state->getValue(['foo', 'bar']);

$values = $form\_state->getValues();

$form\_state->setValue('foo', array('bar'));

$form\_state->setValue(['bar', 'baz'], 'foo');

if ($form\_state->hasValue('foo')) {

}

if ($form\_state->hasValue(['foo', 'bar'])) {

}

if ($form\_state->isValueEmpty('foo')) {

}

if ($form\_state->isValueEmpty(['foo', 'bar'])) {

}

Before:

$form\_state['rebuild'] = TRUE;

$form\_state['rebuild'] = FALSE;

if (!empty($form\_state['rebuild'])) {

}

After:

$form\_state->setRebuild();

$form\_state->setRebuild(FALSE);

if ($form\_state->isRebuilding()) {

}

Before:

$form\_state['build\_info']['callback\_object'] = $callback;

$callback = $form\_state['build\_info']['callback\_object'];

After:

$form\_state->setFormObject($callback);

$callback = $form\_state->getFormObject();

Before:

$form\_state['always\_process'] = TRUE;

$form\_state['always\_process'] = FALSE;

if (!empty($form\_state['always\_process'])) {

}

After:

$form\_state->setAlwaysProcess();

$form\_state->setAlwaysProcess(FALSE);

if ($form\_state->getAlwaysProcess()) {

}

Before:

$buttons = $form\_state['buttons'];

$form\_state['buttons'] = $buttons;

After:

$buttons = $form\_state->getButtons();

$form\_state->setButtons($buttons);

Before:

$form\_state['cache'] = TRUE;

$form\_state['cache'] = FALSE;

$form\_state['no\_cache'] = TRUE;

if (!empty($form\_state['cache']) && empty($form\_state['no\_cache'])) {

}

After:

$form\_state->setCached();

$form\_state->setCached(FALSE);

$form\_state->disableCache();

if ($form\_state->isCached()) {

}

Before:

$form\_state['executed'] = TRUE;

if (!empty($form\_state['executed'])) {

}

After:

$form\_state->setExecuted();

if ($form\_state->isExecuted()) {

}

Before:

$groups = &$form\_state['groups'];

$form\_state['groups'] = $groups;

After:

$groups = &$form\_state->getGroups();

$form\_state->setGroups($groups);

Before:

$form\_state['has\_file\_element'] = TRUE;

$form\_state['has\_file\_element'] = FALSE;

if (!empty($form\_state['has\_file\_element'])) {

}

After:

$form\_state->setHasFileElement();

$form\_state->setHasFileElement(FALSE);

if ($form\_state->hasFileElement()) {

}

Before:

$limit\_validation\_errors = $form\_state['limit\_validation\_errors'];

$form\_state['limit\_validation\_errors'] = $limit\_validation\_errors;

After:

$limit\_validation\_errors = $form\_state->getLimitValidationErrors();

$form\_state->setLimitValidationErrors($limit\_validation\_errors);

Before:

$form\_state['method'] = 'get';

if ($form\_state['method'] == 'get') {

}

After:

$form\_state->setMethod('get');

if ($form\_state->isMethodType('get')) {

}

Before:

$form\_state['must\_validate'] = TRUE;

$form\_state['must\_validate'] = FALSE;

if (!empty($form\_state['must\_validate'])) {

}

After:

$form\_state->setValidationEnforced();

$form\_state->setValidationEnforced(FALSE);

if ($form\_state->isValidationEnforced()) {

}

Before:

$form\_state['process\_input'] = TRUE;

$form\_state['process\_input'] = FALSE;

if (!empty($form\_state['process\_input'])) {

}

After:

$form\_state->setProcessInput();

$form\_state->setProcessInput(FALSE);

if ($form\_state->isProcessingInput()) {

}

Before:

$form\_state['programmed'] = TRUE;

$form\_state['programmed'] = FALSE;

if (!empty($form\_state['programmed'])) {

}

After:

$form\_state->setProgrammed();

$form\_state->setProgrammed(FALSE);

if ($form\_state->isProgrammed()) {

}

Before:

$form\_state['programmed\_bypass\_access\_check'] = TRUE;

$form\_state['programmed\_bypass\_access\_check'] = FALSE;

if (!empty($form\_state['programmed\_bypass\_access\_check'])) {

}

After:

$form\_state->setProgrammedBypassAccessCheck();

$form\_state->setProgrammedBypassAccessCheck(FALSE);

if ($form\_state->isBypassingProgrammedAccessChecks()) {

}

Before:

$form\_state['submitted'] = TRUE;

if (!empty($form\_state['submitted'])) {

}

After:

$form\_state->setSubmitted();

if ($form\_state->isSubmitted()) {

}

Before:

$form\_state['validation\_complete'] = TRUE;

$form\_state['validation\_complete'] = FALSE;

if (!empty($form\_state['validation\_complete'])) {

}

After:

$form\_state->setValidationComplete();

$form\_state->setValidationComplete(FALSE);

if ($form\_state->isValidationComplete()) {

}

Before:

$build\_info = $form\_state['rebuild\_info'];

$form\_state['rebuild\_info'] = $build\_info;

$form\_state['rebuild\_info']['copy'] = $info;

$info = $form\_state['rebuild\_info']['copy'];

After:

$build\_info = $form\_state->getRebuildInfo();

$form\_state->setRebuildInfo($build\_info);

$form\_state->addRebuildInfo('copy', $info);

$info = $form\_state->getRebuildInfo()['copy'];

Before:

$submit\_handlers = $form\_state['submit\_handlers'];

$form\_state['submit\_handlers'] = $submit\_handlers;

After:

$submit\_handlers = $form\_state->getSubmitHandlers();

$form\_state->setSubmitHandlers($submit\_handlers);

Before:

$validate\_handlers = $form\_state['validate\_handlers'];

$form\_state['validate\_handlers'] = $validate\_handlers;

After:

$validate\_handlers = $form\_state->getValidateHandlers();

$form\_state->setValidateHandlers($validate\_handlers);

Before:

$triggering\_element = &$form\_state['triggering\_element'];

$form\_state['triggering\_element'] = $triggering\_element;

After:

$triggering\_element = &$form\_state->getTriggeringElement();

$form\_state->setTriggeringElement($triggering\_element);

**Impacts:**

Module developers

Updates Done (doc team, etc.)

**Online documentation:**

Not done

**Theming guide:**

Not done

**Module developer documentation:**

Not done

**Examples project:**

Not done

**Coder Review:**

Not done

**Coder Upgrade:**

Not done

**Other:**

Other updates done

* [Add new comment](https://www.drupal.org/comment/reply/2310411#comment-form)

Comments

[](https://www.drupal.org/u/jbrown)

[**This is not correct:**](https://www.drupal.org/node/2310411#comment-9145843)

[jbrown](https://www.drupal.org/u/jbrown)*commented 3 years ago*

Note this is not correct:

$storage = &$form\_state->getStorage();

$form\_state->setStorage($storage);

Either get $storage as a & reference, or set it once you are finished. It does not make sense to do both.

--  
Jonathan Brown  
<http://jonathanpatrick.me/>

* [reply](https://www.drupal.org/comment/reply/2310411/9145843)

[](https://www.drupal.org/u/timplunkett)

[**These are not functional**](https://www.drupal.org/node/2310411#comment-9168099)

[tim.plunkett](https://www.drupal.org/u/timplunkett)*commented 3 years ago*

These are not functional examples. Just showing what methods are available, and where appropriate, that some getters return by reference.

* [reply](https://www.drupal.org/comment/reply/2310411/9168099)

[](https://www.drupal.org/u/radamiel)

[**How to check which button is clicked...**](https://www.drupal.org/node/2310411#comment-10585980)

[radamiel](https://www.drupal.org/u/radamiel)*commented 2 years ago*

public function submitForm(array &$form, FormStateInterface $form\_state) {

$clicked\_button = &$form\_state->getTriggeringElement()['#parents'][0];

if ($clicked\_button == 'save') {

// ...

} elseif ($clicked\_button == 'delete') {

// ...

}

}

* [reply](https://www.drupal.org/comment/reply/2310411/10585980)

[](https://www.drupal.org/u/skaught)

[**#validate and #submit still**](https://www.drupal.org/node/2310411#comment-11033327)

[SKAUGHT](https://www.drupal.org/u/skaught)*commented about a year ago*

#validate and #submit still exist.. rather then odd logic loops in one the submit.

....

class myForm extends FormBase {

public function buildForm(array $form, \Drupal\Core\Form\FormStateInterface $form\_state) {

$form['actions'] = ['#type' => 'actions'];

$form['actions']['submit'] = [

'#type' => 'submit',

'#value' => t('Save Changes'),

'#validate' => array('::duplicate\_check', '::empty\_check'),

];

$form['actions']['alpha-sort'] = array(

'#type' => 'submit',

'#value' => t('Sort Alpabetically') . '°',

'#submit' => array('::alpha\_sort'),

'#validate' => array('::duplicate\_check', '::empty\_check'),

);

}

public function duplicate\_check(array &$form, FormStateInterface $form\_state) {

// do stuff..

}

}

now that this is within it's own class the:: is used to calls the follow up function. OOP tricks.

(: also, as your sample is a delete situation, check into [Removed confirm\_form() in favor of \Drupal\Core\Form\ConfirmFormBase](https://www.drupal.org/node/1945416)to verify delete operations. better to be safe than sorry.

* [reply](https://www.drupal.org/comment/reply/2310411/11033327)

[](https://www.drupal.org/u/nithinkolekar)

[**no follow button :(.**](https://www.drupal.org/node/2310411#comment-11596657)

[nithinkolekar](https://www.drupal.org/u/nithinkolekar)*commented about a year ago*

no follow button :(.

hence commenting..

* [reply](https://www.drupal.org/comment/reply/2310411/11596657)

[](https://www.drupal.org/u/rob230)

[**If only it was this simple**](https://www.drupal.org/node/2310411#comment-12013317)

[Rob230](https://www.drupal.org/u/rob230)*commented 6 months ago*

Before: $foo = $form\_state['values']['foo']

After: $foo = $form\_state->getValue('foo');

Actually, in Drupal 7 $form\_state['values']['foo'] would have returned the submitted value. In Drupal 8 it appears to be an array. You have to access it like: $foo = $form\_state->getValue('foo')[0]['value']. For a multivalue field as well as the values themselves it will have one blank item, and an "Add another item" object, which means stepping through the array to look at the values is ugly.

* [reply](https://www.drupal.org/comment/reply/2310411/12013317)

[](https://www.drupal.org/u/yash_khandelwal)

[**Hello,**](https://www.drupal.org/node/2310411#comment-12092096)

[yash\_khandelwal](https://www.drupal.org/u/yash_khandelwal)*commented 4 months ago*

Hello,

I am having a problem with port below code to Drupal 8.

$form\_state['field\_deltas'][] = count($form\_state['field\_deltas']) > 0 ?max($form\_state['field\_deltas'])+1: 0;

Please suggest the solution for this.

* [reply](https://www.drupal.org/comment/reply/2310411/12092096)

[](https://www.drupal.org/u/pankajxenix)

[**setValue Not working properly .**](https://www.drupal.org/node/2310411#comment-12095262)

[pankajxenix](https://www.drupal.org/u/pankajxenix)*commented 4 months ago*

Hi,  
When I tried to alter a field value it replace first latter of string to 0.

$form\_state->setValue('foo', array('bar'));

That means result of this code is 0ar.Also get error  
Warning: Illegal string offset '\_original\_delta' in Drupal\Core\Field\WidgetBase->extractFormValues() (line 361 of core/lib/Drupal/Core/Field/WidgetBase.php).

I am using this in custom validation of form.

* [reply](https://www.drupal.org/comment/reply/2310411/12095262)

[](https://www.drupal.org/u/cocorodeo)

[**Same issue with setValue()**](https://www.drupal.org/node/2310411#comment-12148633)

[cocorodeo](https://www.drupal.org/u/cocorodeo)*commented 3 months ago*

Hi pankajdrync,

I'm having the same issue on custom entity submitForm,  
$from\_state->setValue('foo', ['value'])  
is generate warning messages "Warning: Illegal string offset '\_original\_delta' ..."

The value is correctly saved but I get warnings...

Did you find a solution to avoid these warnings ?