

DrupalCon

SEATTLE 2019
APRIL 8-12





DrupalCon
SEATTLE 2019
APRIL 8-12

New Wave Module Development

Kris “EclipseGc” Vanderwater



Kris Vanderwater

Sr. Software Engineer @ Acquia

 @EclipseGc

- Drupal Development 13+ years
- CTools Co-Maintainer
- Multi-year focus on Page Layout
- Drupal 8 Contributor
- Co-Author: Drupal 8 Plugin System
- Major contributor to Layout Builder
- Technical Architect ContentHub 2.0





DrupalCon
SEATTLE 2019
APRIL 8-12

Topics



DrupalCon
SEATTLE 2019
APRIL 8-12

Topics

- Drupal 7 & Earlier Development



DrupalCon
SEATTLE 2019
APRIL 8-12



Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements
- Practical Application



DrupalCon
SEATTLE 2019
APRIL 8-12

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements
- Practical Application
- Question/Answer



Drupal 7 & Earlier Development





“

It's OK to figure out murder mysteries,
but you shouldn't need to figure out
code. You should be able to read it.

”

Steve McConnell



DrupalCon
SEATTLE 2019
APRIL 8-12

Drupal Hooks



Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named `foo_bar()`, where "foo" is the name of the module (whose filename is thus `foo.module`) and "bar" is the name of the hook.



Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named `foo_bar()`, where "foo" is the name of the module (whose filename is thus `foo.module`) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named `foo_bar()`, where "foo" is the name of the module (whose filename is thus `foo.module`) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.



Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named `foo_bar()`, where "foo" is the name of the module (whose filename is thus `foo.module`) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.
- Hooks are procedural functions (No dependency injection)

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named `foo_bar()`, where "foo" is the name of the module (whose filename is thus `foo.module`) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.
- Hooks are procedural functions (No dependency injection)
- `drupal_static()`...



DrupalCon
SEATTLE 2019
APRIL 8-12

Drupal Hooks

```
function foo_bar($arg1, $arg2) {  
  if ($arg1 === 'foo') {  
    // do stuff.  
  }  
}
```



Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  switch($arg1) {
    case 'foo':
      // do stuff
      break;
    case 'bar':
      // do other stuff
      break;
  }
}
```



Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  if ($arg1 && $arg1 === 'foo' && $arg2 === 'bar') {
    // do stuff
  }
  if ($arg2 === 'baz') {
    // do other stuff
  }
  elseif ($arg2 === 'foobaz') {
    // some other stuff
  }
}
```



Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  $foo = &drupal_static(__FUNCTION__, []);
  if (empty($foo)) {
    // Do something expensive and store it in $foo
  }
  return $foo;
}
```



Drupal Hooks (ALL TOGETHER NOW!)

```
function foo_bar($arg1, $arg2) {
  $foo = &drupal_static(__FUNCTION__, []);
  if (empty($foo)) {
    // Get a service!
    /** @var \Drupal\foo\MyServiceInterface $service */
    $service = \Drupal::service('get.some.service');
    // Do something expensive and store it in $foo
  }
  return $foo;
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Drupal Hooks



DrupalCon
SEATTLE 2019
APRIL 8-12

Drupal Hooks

- Hooks started life manipulating strings (HTML output).



Drupal Hooks

- Hooks started life manipulating strings (HTML output).
- Alter hooks are a completely separate thing that pass references.



Drupal Hooks

- Hooks started life manipulating strings (HTML output).
- Alter hooks are a completely separate thing that pass references.
- Alter hooks only support a limited number of references at a time.



Fossils of the Fore-bearers



“

Even the best planning is
not so omniscient as to
get it right the first time.”

Fred Brooks



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub





DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- Content Syndication Service



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7



Acquia ContentHub



- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined

Acquia ContentHub



- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined
- Looks like a stereotypical Drupal module

Acquia ContentHub



- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined
- Looks like a stereotypical Drupal module
- Difficult to maintain



DrupalCon

SEATTLE 2019
APRIL 8-12

Acquia ContentHub



```
ted function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
  $var\Drupal\Core\FieldItemListInterface[] $fields;
  $ids = $entity->getFields();
  // Our field mapping. This maps drupal field types to Content Hub
  // attribute types. $this->getFieldTypeMapping($entity);
  // ignore the entity ID and revision ID.
  $excluded_fields = $this->excludedProperties($entity);
  // continue if $entity->hasField($field) as $entity->hasField($field) returns true if the user does not
  // have access to view it or the current user does not
  // (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
  continue;

  // Get the plain version of the field in regular json.
  $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
  $entity->addValue($name, $serialized_field);

  // Given that vocabularies are configuration entities, they are not
  // supported in Content Hub. Instead we use the type mapping name
  // ($name === 'vid' && $entity->getEntityType() === 'taxonomy_term') {
  // Initialize vocabulary attribute if it doesn't exist yet.
  if ($entity->getEntityType() === 'taxonomy_term') {
    $attribute->setAttributeValue('vocabulary', $entity->getVocabulary());
    $contenthub_entity->setAttribute('vocabulary', $attribute);
  }
  else {
    $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
  }
  continue;

  // To make it work with Paragraphs, we are converting the field
  // parent_id to parent_uuid because Content Hub cannot deal with
  // entity_id information.
  $attribute = new Attribute(Attribute::TYPE_STRING);
  $parent_type = $items[0].value['parent_type'];
  $parent_id = $items[0].value['parent_id'];
  $parent_type->getValue()['value'];
  $load($parent_id);
  $parent_uuid = $parent_type->parent_id;
  $attribute->setValue($parent_uuid, $langcode);
  $contenthub_entity->setAttribute('parent_uuid', $attribute);
  continue;

  // Try to map it to a known field type.
  // Go to the fallback data type when the field type is not known.
  // $type = $type_mapping[$name];
  $type = $type_mapping[$name];
  if (isset($type_mapping[$field_type])) {
    $type = $type_mapping[$field_type];
  }
  else {
    $type = $type_mapping[$name];
  }

  // ($type == NULL) {
  continue;

  // $field[] instanceof EntityReferenceFieldItemListInterface) {
  // Get taxonomy parent terms
  if ($entity->getEntityType() === 'taxonomy_term') {
    $storage = $entity->getTypeManager()->getStorage('taxonomy_term');
    $referenced_entities = $storage->loadParents($entity->id());
  }
  else {
    $referenced_entities = $entity->referencedEntities();
  }

  $values[$langcode] = [];
  foreach ($referenced_entities as $key => $referenced_entity) {
    // In the case of images/files, etc... we need to add the assets.
    $file_types = [
      'image',
      'file',
      'video',
    ];
    $type_names = [
      'bundle',
    ];
    // Special case for type as we do not want the reference for the
    // bundle field which stores a media bundle configuration entity UUID.
    if (in_array($name, $type_names, TRUE) && $referenced_entity instanceof ConfigEntityBase) {
      $values[$langcode][$key] = $referenced_entity->id();
    }
    elseif (in_array($field_type, $file_types)) {
      $uid_token = file_create_url($referenced_entity->getFileUri());
      $asset = new Asset();
      $asset->setUrl($asset->url);
      $asset->setReplaceToken($uid_token);
      $contenthub_entity->addAsset($asset);
    }
    // Now add the value
    // from the file entity including the "alt" and "title" attributes
    $data = [
      'alt' => isset($items[$key]['alt']) ? $items[$key]['alt'] : '',
      'title' => isset($items[$key]['title']) ? $items[$key]['title'] : '',
      'target_uuid' => $uid_token,
    ];
    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
  }
  else {
    $values[$langcode][] = $referenced_entity->uuid();
  }
}

// If there's nothing in this field, just set it to NULL.
if ($items == NULL) {
  $values[$langcode] = NULL;
}
else {
  // Only if it is a link type
  if ($field->isLinkField()) {
    $entity->normalizeItems($items);
  }
  // Loop over the items to get the values for each field.
  foreach ($items as $item) {
    // TODO: Find a better solution for this.
    if (isset($item['attributes'])) {
      unset($item['attributes']);
    }
    $keys = is_array($item) ? array_keys($item) : [];
    $value = $item['value'];
    if (count($keys) == 1 && isset($item['value'][0])) {
      $value = $item['value'][0];
    }
    else {
      if ($field instanceof PathFieldItemList) {
        $item = $field->first();
        $item['source'] = '';
        $value = json_encode($item, JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
      }
      $values[$langcode][] = $value;
    }
  }
}

$attribute = new Attribute($type);
try {
  $args['type'] = $type;
  $message = new FormattableMarkup('No type could be registered for %type.', $args);
  throw new ContenthubException($message);
}
catch (\Exception $e) {
  $attribute->setAttribute('array', $values);
  $value = array_pop($values[$langcode]);
  $attribute->setValue($value, $langcode);

  // If attribute exists already, append to the existing values.
  if (!empty($contenthub_entity->getAttribute($name))) {
    $existing_attribute = $contenthub_entity->getAttribute($name);
    $existing_attribute->appendAttribute($existing_attribute, $attribute->getValues());
    $attribute = $existing_attribute;
  }

  // Add it to our contenthub entity
  $contenthub_entity->setAttribute($name, $attribute);

  // Allow alterations of the CDF to happen.
  $text['language'] = $entity->language();
  $entity->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
  $url = $entity->url();
  if (empty($contenthub_entity->getAttribute('url'))) {
    $entity->getEntityType();
    if ($entity->getEntityType()) {
      $value = file_create_url($entity->getFileUri());
      $contenthub_entity->setAttribute('_filepath', $value);
      $contenthub_entity->setAttribute('filepath', $value);
      break;
    }
    default:
    if ($entity->url() && $entity->hasLinkTemplate('canonical')) {
      $url = $entity->url();
      $value = $url->toString();
      $url->setAbsolute(TRUE);
      $value = $url->toString();
      break;
    }
    if (isset($value)) {
      $attribute = new Attribute(Attribute::TYPE_STRING);
      $contenthub_entity->setAttribute('url', $attribute);
      $contenthub_entity->setAttribute('url', $url);
      $contenthub_entity->setAttribute('url', $value);
      $contenthub_entity->setAttribute('url', $langcode);
    }
  }
}

return $contenthub_entity;
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



■ 1 method

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. $entity->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($excluded_fields as $field) {
        if (!ContentHubEntity::hasField($entity, $field)) {
            continue;
        }
        // Get the plain version of the field in regular json.
        $serialized_field =&$this->getSerializer()->normalize($field, 'json', $context);
        $item = $serialized_field;
        // Given that vocabulary hub are configuration entities they are not
        // supported in ContentHub. Instead we use the vocabulary machine name
        if ($field->getEntityTypeId() == 'taxonomy_term') {
            if ($contenthub_entity->getAttribute('vocabulary')) {
                $attribute->setValues($entity->target_id, $langcode);
                $contenthub_entity->setAttribute('vocabulary', $attribute);
            } else {
                $contenthub_entity->setAttributeValue('vocabulary', $entity[0]['target_id'], $langcode);
            }
            continue;
        }
        // To make it work with Paragraphs we are converting the field
        // to its parent type because Content Hub cannot deal with
        if ($field->getParentId() && $entity->getEntityTypeId() == 'paragraph') {
            $parent_id = $entity[0]['value'][0][0];
            $parent_type = $entity[0]['value'][0][0]->getValue()['target_type'];
            $parent_attributes = $entity[0]['value'][0][0]->getStorage($parent_type)->load($parent_id);
            $parent_attributes->setAttribute('parent_id', $langcode);
            $contenthub_entity->setAttribute('parent_id', $parent_attributes);
            continue;
        }
        if ($field->getEntityType() == 'media') {
            $contenthub_entity->setAttribute('bundle', $attribute);
            continue;
        }
        // Try to map it to a known field type.
        // Go to the fallback datatype when the field type is not known.
        if ($entity->getEntityType() != $field->getEntityType()) {
            $type = $entity->getEntityType();
            elseif ($type->getMapping($field->getType())) {
                $type = $type->getMapping($field->getType());
            }
            if ($type == NULL) {
                $value_field = $entity->getInstanceof(EntityReferenceFieldItemListInterface);
                if ($value_field->getTaxonomyParentTerm() == 'taxonomy_term') {
                    $referenced_entities = $storage->loadParents($entity->id());
                    $entity->referenced_entities = $referenced_entities;
                }
                $values[$langcode] = [];
                foreach ($referenced_entities as $key => $referenced_entity) {
                    $file_types = $entity->getTypesOfImagesFiles();
                    $file_type = $entity->getMediaFile($key);
                    $type_name = [
                        'image',
                        'video',
                    ];
                    // Special case for type as we do not want the reference for the
                    // bundle which is the type of the media entity has a UUID.
                    if (is_array($name) && $referenced_entity->isinstanceof(ConfigEntityBase)) {
                        $values[$langcode][] = $referenced_entity->id();
                    } elseif (in_array($field->getType(), $file_types)) {
                        $dataset = new AssetDataset();
                        $dataset->newAssetset($referenced_entity->getFileUri());
                        $contenthub_entity->addDataset($dataset);
                    }
                    // Now add the value, including the "alt" and "title" attributes
                    $data[] = isset($entity[$key][$field->getName()]) ? $entity[$key][$field->getName()] : '';
                    $target_url = $entity[$key][$field->getName()];
                    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                }
                else {
                    $values[$langcode][] = $referenced_entity->uuid();
                }
            }
            else {
                if (there's nothing in this field, just set it to NULL.
                $values[$langcode] = NULL;
            }
            else {
                if ($entity->isLinkFieldHandler) {
                    $entity->linkFieldHandler->load($field)->validate();
                }
                // Loop over the items to get the values for each field.
                foreach ($entity->getItems() as $item) {
                    if (isset($item->attributes)) {
                        $item->unset($item->attributes);
                    }
                    if ($entity->getEntityType() == 'entityreference') {
                        $key = $entity->key();
                        $key_value = $item[$key];
                        $value = $item[$key];
                    }
                    elseif ($field->getInstanceof(PathFieldItemList)) {
                        $item['tid'] = '';
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                    }
                    $values[$langcode][] = $value;
                }
            }
            try {
                $attribute = new Attribute($type);
            } catch (\Exception $e) {
                $message = $e->getMessage();
                throw new ContentHubException($message);
            }
            if ($attribute->type == 'array') {
                $value = array_get($values[$langcode]);
            } else {
                $value = array_pop($values[$langcode]);
            }
            $attribute->setValue($value, $langcode);
            // If attribute exists already, append to the existing values.
            $existing_attributes = $contenthub_entity->getAttribute($name);
            $attribute->setAttribute('existing_attributes', $existing_attributes);
            // Add it to our contenthub entity.
            $contenthub_entity->setAttribute($name, $attribute);
            // Allow alterations of the CDF to happen.
            $entity->entityType->setEntityType($entity->getEntityType());
            $entity->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
            // Adds the entity URL to CDF.
            if (empty($contenthub_entity->getAttribute('url'))) {
                $entity->createUrl($entity->getFileUri(), TYPE_STRING);
                $contenthub_entity->setAttribute('_filepath', $entity->getFilepath());
                $entity->setAttribute('filepath', $entity->getFilepath());
                default:
                    if ($entity->isLink()) {
                        $uri = $entity->getLinkTemplate('canonical');
                        $value = $uri->getUri();
                    }
                    break;
                if (isset($value)) {
                    $entity->setAttribute($name, $value);
                    $entity->setAttribute($name, $value, $langcode);
                }
            }
            return $contenthub_entity;
        }
    }
}
```



DrupalCon

SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
 - 229 lines of code

```
function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $var \Drupal\Core\Field\ItemListInterface[] $fields;
    $fields = $entity->getFields();
    // Set our field mapping. This maps drupal field types to Content Hub
    // attribute types. $this->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_comes_here = $entity->getExcludedProperties();
    $excluded_fields = $entity->getExcludedFields();
    // Continue if this is an excluded field or the current user does not
    // have access to view it (FieldDefinition->getName(), $excluded_fields) || !$field->access('view', $context['account']));
    continue;

    // Get the plain version of the field in regular json.
    $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
    $entity->addValue($name, $value);
    $entity->addValue($name, $value);
    $entity->addValue($name, $value);

    // Given that in Content Hub instead we use the vocabulary machine name
    // as mechanism to syndicate and import them in the right taxonomy term.
    if ($name === 'vid' && $entity->getEntityType() === 'taxonomy_term') {
        $initial_value = $entity->getInitialValue();
        $entity->setAttribute('target_id', $initial_value);
        $entity->addValue($name, $initial_value);
        $entity->addValue($name, $initial_value);
        $entity->addValue($name, $initial_value);
    }

    // To make it work with Paragraphs we are converting the field
    // parent_id information because Content Hub cannot deal with
    // entities with parent_id.
    if ($name === 'parent_id' && $entity->getEntityType() === 'paragraph') {
        $attribute = new Attribute(Attribute::TYPE_STRING);
        $parent_id = $entity->getInitialValue();
        $parent_type = $entity->getInitialValue();
        $parent_type->entityTypeManager->getStorage($parent_type)->load($parent_id);
        $parent_uuid = $parent_type->id();
        $attribute->setValue($parent_uuid, $langcode);
        $contenthub_entity->setAttribute('vocabulary', $attribute);
        continue;
    }

    // Try to map it to a known field type.
    if ($name == 'bundle' && $entity->getEntityType() == 'media') {
        $attribute = new Attribute(Attribute::TYPE_ARRAY_STRING);
        $attribute->setValue($entity->bundle, $langcode);
        $contenthub_entity->setAttribute('bundle', $attribute);
        continue;
    }

    // Try to map it to a known field type.
    if ($name == 'file' && $entity->getEntityType() == 'file') {
        $type = $type_mapping[$name];
        if (isset($type_mapping[$name])) {
            $type = $type_mapping[$name];
        } else {
            $type = $type_mapping['fallback'];
        }
        $entity->addValue($name, $type);
        $entity->addValue($name, $type);
        $entity->addValue($name, $type);
    }

    // Special case for type as we do not want the reference for the
    // bundle. In addition to the type field a media entity has a
    // target_uid which stores the media entity's UUID.
    if ($entity->getEntityType() == 'media') {
        $referenced_entities = $entity->getReferencedEntities();
        foreach ($referenced_entities as $key => $referenced_entity) {
            $values[$langcode][] = [
                'type' => $type,
                'target_id' => $referenced_entity->id(),
                'target_type' => $referenced_entity->type(),
                'target_label' => $referenced_entity->label(),
                'target_url' => $referenced_entity->url(),
                'target_replacement_token' => $referenced_entity->replacementToken(),
                'target_is_new' => $referenced_entity->isExternal()
            ];
        }
    }

    // If there's nothing in this field, just set it to NULL.
    if ($entity->getEntityType() == 'link') {
        $entity->addValue($name, NULL);
    } else {
        $entity->addValue($name, $value);
    }

    // Loop over the items to get the values for each field.
    foreach ($entity->getItems() as $item) {
        $entity->addValue($name, $value);
    }

    // Add it to our ContentHub entity.
    $contenthub_entity->setAttribute($name, $value);

    // Allow alterations of the CDF to happen.
    $entity->language->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
}

// If attribute exists already, append to the existing values.
if (!empty($contenthub_entity->getAttribute($name))) {
    $existing_attribute = $contenthub_entity->getAttribute($name);
    $existing_attribute->appendAttribute($entity->getattribute($name));
    $contenthub_entity->setAttribute($name, $existing_attribute);
}

// Add it to our ContentHub entity.
$entity->language->alter('acquia_contenthub_cdf', $contenthub_entity, $context);

// Set the entity URL to CDF.
$entity->language->empty($contenthub_entity->getAttribute('url')));
$entity->language->getEntityType();
case 'file':
    $value = file_create_url($entity->getFileUri());
    $contenthub_entity->setAttribute('_filepath', $value);
    $contenthub_entity->setAttribute($name, $value);
    break;
default:
    if ($entity->language->isNew() && $entity->hasLinkTemplate('canonical')) {
        $url = $entity->language->tour();
        $url->setAbsolute(TRUE);
        $value = $url->toString();
        break;
    }
    if (isset($value)) {
        $entity->language->newAttribute(Attribute::TYPE_STRING);
        $entity->language->setAttribute('url', $value);
        $entity->language->setValue($value, $langcode);
    }
}

// Set the entity URL to CDF.
$entity->language->empty($contenthub_entity->getAttribute('url')));
$entity->language->getEntityType();
case 'file':
    $value = file_create_url($entity->getFileUri());
    $contenthub_entity->setAttribute('_filepath', $value);
    $contenthub_entity->setAttribute($name, $value);
    break;
default:
    if ($entity->language->isNew() && $entity->hasLinkTemplate('canonical')) {
        $url = $entity->language->tour();
        $url->setAbsolute(TRUE);
        $value = $url->toString();
        break;
    }
    if (isset($value)) {
        $entity->language->newAttribute(Attribute::TYPE_STRING);
        $entity->language->setAttribute('url', $value);
        $entity->language->setValue($value, $langcode);
    }
}

// Set the entity URL to CDF.
$entity->language->empty($contenthub_entity->getAttribute('url')));
$entity->language->getEntityType();
case 'file':
    $value = file_create_url($entity->getFileUri());
    $contenthub_entity->setAttribute('_filepath', $value);
    $contenthub_entity->setAttribute($name, $value);
    break;
default:
    if ($entity->language->isNew() && $entity->hasLinkTemplate('canonical')) {
        $url = $entity->language->tour();
        $url->setAbsolute(TRUE);
        $value = $url->toString();
        break;
    }
    if (isset($value)) {
        $entity->language->newAttribute(Attribute::TYPE_STRING);
        $entity->language->setAttribute('url', $value);
        $entity->language->setValue($value, $langcode);
    }
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
- 229 lines of code
- 4 separate entity type checks

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. $entity->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($entity->getFields() as $field) {
        if (!in_array($field, $excluded_fields) && $entity->access('view', $context['account'])) {
            // Get the plain version of the field in regular json.
            $serialized_field =& $this->getSerializer()->normalize($field, 'json', $context);
            $item = $serialized_field;
            // Given that vocabulary hub are configuration entities they are not
            // supported in ContentHub. Instead we use the vocabulary machine name
            if ($field->getEntityTypeId() == 'taxonomy_term') {
                if ($contenthub_entity->getAttribute('vocabulary')) {
                    $attribute->setValues($entity->getTargetId(), $langcode);
                    $contenthub_entity->setAttribute('vocabulary', $attribute);
                } else {
                    $contenthub_entity->setAttributeValue('vocabulary', $entity[0]['target_id'], $langcode);
                }
            }
            // To make it work with Paragraphs we are converting the field
            // to its parent type because Content Hub cannot deal with
            if ($field->getParentId() && $entity->getEntityType() == 'paragraph') {
                $parent_id = $entity[0]['value'][0][1];
                $parent_type = $entity[0]['value'][0][1];
                $parent_type->getStorage($parent_type)->load($parent_id);
                $attribute->setValues($parent_id, $langcode);
                $contenthub_entity->setAttribute('parent_id', $attribute);
            }
            // Try to map it to a known field type.
            // Go to the fallback datatype when the field type is not known.
            if ($entity->getEntityType() == 'node') {
                $type = $entity->getEntityTypeId();
                elseif ($type == $entity->getEntityType()) {
                    $type = $entity->getEntityType();
                } else {
                    $type = $entity->getEntityType();
                }
                if ($type == NULL) {
                    $values[$langcode] = [];
                    if ($entity->instanceof EntityReferenceFieldItemListInterface) {
                        if ($entity->getEntityTypeId() == 'taxonomy_term') {
                            $referenced_entities = $storage->loadParents($entity->id());
                            $entity->referenced_entities = $referenced_entities;
                        }
                        $values[$langcode] = [];
                        foreach ($referenced_entities as $key => $referenced_entity) {
                            $file_types = [
                                'image',
                                'video',
                            ];
                            $type_name = [
                                'image',
                                'video',
                            ];
                            // Special case for type as we do not want the reference for the
                            // type added to the media entity. We need to create a new instance of ConfigEntityBase {
                            if (is_array($name) && $referenced_entity->id() != $entity->id()) {
                                $entity->id();
                            }
                            elseif (in_array($field->type, $file_types)) {
                                $entity->id();
                            }
                            $asset->newAssetSet($entity->id());
                            $contenthub_entity->entityTokenAdd($asset);
                            // Now add the value, including the "alt" and "title" attributes
                            $data[] = isset($entity[$key]['alt']) ? $entity[$key]['alt'] : '',
                            $data[] = isset($entity[$key]['title']) ? $entity[$key]['title'] : '';
                            $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                        }
                    } else {
                        $values[$langcode][] = $referenced_entity->uuid();
                    }
                }
            }
            if ($values[$langcode] == NULL) {
                $values[$langcode] = [];
            }
            if ($entity->isLinkType()) {
                $entity->load($field->validate());
                $entity->normalise();
                // Loop over the items to get the values for each field.
                foreach ($entity->items as $item) {
                    if (isset($item['attributes'])) {
                        $entity->unset($item['attributes']);
                    }
                    if (empty($item['key'])) {
                        $entity->key($item['value']);
                    }
                    if ($entity->instanceof PathFieldItemList) {
                        $entity->id();
                        $entity->value();
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                    }
                    $values[$langcode][] = $value;
                }
            }
            if ($entity->newFormatableMarkup('No type could be registered for type.', $args)) {
                throw new ContentHubException($message);
            }
            if ($entity->type == 'array') {
                $entity->setValues($values);
            } else {
                $entity->arrayPop($values[$langcode]);
                $entity->setValues($value, $langcode);
            }
            // If attribute exists already, append to the existing values.
            $existing_attributes = $contenthub_entity->getAttribute($name);
            $attribute->setAttribute('existing_attributes', $existing_attributes);
            // Add it to our contenthub entity.
            $contenthub_entity->addAttribute($name, $attribute);
            // Allow alterations of the CDF to happen.
            $entity->entity();
            $entity->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
            // Adds the entity URL to CDF.
            if (empty($contenthub_entity->getAttribute('url'))) {
                $entity->fileCreateUrl($entity->getFileUri(), TYPE_STRING);
                $contenthub_entity->setAttribute('_filepath', $filepath->setValue($entity->getFileUri()));
            }
            default:
                if ($entity->type == 'entity') {
                    $entity->entity();
                    $entity->value();
                    $value = $entity->canonical();
                    break;
                }
                if (isset($value)) {
                    $entity->Attribute($attribute, TYPE_STRING);
                    $entity->setAttribute('url', $url->setValue($value, $langcode));
                }
            }
        }
    }
    return $contenthub_entity;
}
```



DrupalCon

SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
 - 229 lines of code
 - 4 separate entity type checks
 - Implies a need for custom logic

```

ected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    $type_mapping = $this->getFieldTypeMapping($entity);
    // Ignore the entity ID and revision ID.
    $excluded_comes_here = $this->excludedProperties($entity);
    $excluded_fields = $this->excludedFields($entity);
    // Continue if either an excluded field or the current user does not
    if (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
        continue;
    }
    // Get the plain version of the field in regular json.
    if ($name === 'metatag') {
        $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
    }
    else {
        $serialized_field = $field->getValue();
    }
    $items = $serialized_field;
    // Given that vocabularies are configuration entities, they are not
    // supported in Content Hub. Instead we use the vocabulary machine name
    if ($name === 'vid' && $entity->getEntityType() === 'taxonomy_term') {
        if (!isset($contenthub_entity->getAttributes()['vocabulary'])) {
            $attribute->setAttributeValue('target_id', $name);
            $contenthub_entity->setAttribute('vocabulary', $attribute);
        }
        else {
            $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
        }
        continue;
    }
    // To make it work with Paragraphs, we are converting the field
    // parent_id to parent_uuid because Content Hub cannot deal with
    if ($name === 'parent' && $entity->getEntityType() === 'paragraph') {
        $parent_id = new Attribute($attribute->getTypeId());
        $parent_type = $items[0]['value'];
        $parent_type->load($parent_type);
        $parent_uuid = $parent_type->getEntityTypeId();
        $parent_type->load($parent_id);
        $attribute->setValue($parent_uuid, $langcode);
        $contenthub_entity->setAttributeValue('parent_uuid', $attribute);
        continue;
    }
    if ($name === 'bundle' && $entity->getEntityType() === 'media') {
        $attribute->setValue($bundle, $langcode);
        $contenthub_entity->setAttributeValue('bundle', $attribute);
        continue;
    }
    // Try to map it to a known field type.
    // Go to the fallback datatype when the field type is not known.
    if (isset($type_mapping[$name])) {
        $type = $type_mapping[$name];
    }
    elseif (isset($type_mapping[$field_type])) {
        $type = $type_mapping[$field_type];
    }
    if ($type == NULL) {
        continue;
    }
    $values[$field] = [];
    if ($field instanceof EntityReferenceFieldItemListInterface) {
        // Get taxonomy parent terms
        if ($name === 'parent' && $entity->getEntityType() === 'taxonomy_term') {
            $storage = $entity->getStorage('taxonomy_term');
            $referenced_entities = $storage->loadParents($entity->id());
        }
        else {
            $referenced_entities = $entity->referencedEntities();
        }
        $values[$langcode] = [];
        foreach ($referenced_entities as $key => $referenced_entity) {
            // In the case of images/files, etc... we need to add the assets.
            $file_types = [
                'image',
                'video',
            ];
            $type_names = [
                'type',
                'bundle',
            ];
            // Special case for type as we do not want the reference for the
            // bundle field which stores a media entity has a UUID.
            if (in_array($name, $type_names) && $referenced_entity instanceof ConfigEntityBase) {
                if (in_array($langcode, $values[$langcode])) {
                    $referenced_entity->id();
                }
                elseif (in_array($field_type, $file_types)) {
                    $asset_uris[] = file_create_url($referenced_entity->getFileUri());
                    $asset = new Asset();
                    $asset->setUri($asset_uris);
                    $contenthub_entity->addAsset($asset);
                }
                // Now add the value
                // Notice the entity including the "alt" and "title" attributes
                $data['alt'] = '';
                $data['title'] = '';
                if (isset($items[$key]['alt'])) {
                    $data['alt'] = $items[$key]['alt'];
                }
                if (isset($items[$key]['title'])) {
                    $data['title'] = $items[$key]['title'];
                }
                $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
            }
            else {
                $values[$langcode][] = $referenced_entity->uuid();
            }
        }
    }
    else if (empty($values[$langcode])) {
        $values[$langcode] = NULL;
    }
    else {
        // Only if it is a link type
        if ($field instanceof ContentHubEntityLinkFieldHandler) {
            $item = $field->normalizeItems($values);
        }
        // Loop over the items to get the values for each field.
        foreach ($item as $item) {
            // TODO: Find a better solution for this.
            if (isset($item['attributes'])) {
                unset($item['attributes']);
            }
            $keys = is_array($item) ? array_keys($item) : [];
            if (count($keys) == 1 && isset($item[$value])) {
                $value = $item[$value];
            }
            else {
                if ($field instanceof PathFieldItemList) {
                    $item['source'] = '';
                    $item['target'] = '';
                }
                $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
            }
        }
    }
    try {
        $attribute = new Attribute($type);
    }
    catch (\Exception $e) {
        $message = new FormattableMarkup('No type could be registered for %type.', $args);
        throw new ContentHubException($message);
    }
    if ($attribute->isArray()) {
        $attribute->setValues($values);
    }
    else {
        $value = array_pop($values[$langcode]);
        $attribute->setValue($value, $langcode);
    }
    // If attribute exists already, append to the existing values.
    if (!empty($contenthub_entity->getAttribute($name))) {
        $existing_attribute = $contenthub_entity->getAttribute($name);
        $existing_attribute->append($attribute);
        $attribute = $existing_attribute;
    }
    // Add it to our contenthub entity.
    $contenthub_entity->setAttribute($name, $attribute);

    // Allow alterations of the CDF to happen.
    $context['langcode'] = $langcode;
    $this->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
    // Adds the entity URL to CDF.
    $value = NULL;
    if (empty($contenthub_entity->getAttribute('url'))) {
        global $base_path;
        switch ($entity->getEntityType()) {
            case 'entity':
                $filepath_attribute = new Attribute($entity->getFilePath());
                $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFilePath()));
                break;
            default:
                // Get entity URL
                if ($entity->isNew() && $entity->hasLinkTemplate('canonical')) {
                    $uri = $entity->getLinkTemplate();
                    $uri->setAbsolute(TRUE);
                    $value = $uri->toString();
                }
                break;
        }
        if (isset($value)) {
            $attribute = new Attribute($attribute, $type);
            $contenthub_entity->setAttribute('url', $attribute)->setValue($value, $langcode));
        }
    }
    return $contenthub_entity;
}

```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
- 229 lines of code
- 4 separate entity type checks
- Implies a need for custom logic
- 7 different field name checks

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. $entity->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($excluded_fields as $field) {
        if ($entity->hasField($field) || $entity->isExcludedField($field)) {
            continue;
        }
        $entity->removeField($field);
    }
    // Get the plain version of the field in regular json.
    $serialized_field = $entity->getSerializer()->normalize($entity, 'json', $context);
    $items = $serialized_field['fields'];
    // Given that vocabulary hub are configuration entities they are not
    // supported in ContentHub. Instead we use the vocabulary machine name
    if ($name->isVocabulary($entity->getEntityType()) == 'taxonomy_term') {
        if ($contenthub_entity->getEntityType() == 'contenthub_entity') {
            $attribute->setValues($entity->getTargetId(), $langcode);
            $contenthub_entity->setAttribute('vocabulary', $attribute);
        } else {
            $contenthub_entity->setAttributeValue('vocabulary', $entity[0]['target_id'], $langcode);
        }
        continue;
    }
    // To make it work with Paragraphs we are converting the field
    // id to the parent uuid because Content Hub cannot deal with
    if ($name->isParagraph($entity->getEntityType()) == 'paragraph') {
        $parent_id = $entity[0]['value'][0][0];
        $parent_type = $entity[0]['value'][0][1];
        $parent_type->getValue();
        $parent_type->getStorage($parent_type)->load($parent_id);
        $attribute->setValues($parent_id, $langcode);
        $contenthub_entity->setAttribute('parent_uuid', $attribute);
        continue;
    }
    if ($name == 'bundle' || $entity->getEntityType() == 'media') {
        $contenthub_entity->setAttribute('bundle', $attribute);
        continue;
    }
    // Try to map it to a known field type.
    // Go to the fallback datatype when the field type is not known.
    if ($entity->getEntityType() != 'string') {
        $type = $entity->getEntityType();
        elseif ($entity->getEntityType() == 'string') {
            $type = $entity->getEntityType();
        }
        if ($type == NULL) {
            $value_field instanceof EntityReferenceFieldItemListInterface) {
                if ($name == $parent_id && $entity->getEntityType() == 'taxonomymTerm') {
                    $referenced_entities = $storage->loadParents($entity->id());
                    $entity->referenced_entities = $referenced_entities;
                }
                $values[$langcode] = [];
                foreach ($referenced_entities as $key => $referenced_entity) {
                    $file_types = [
                        'image',
                        'video',
                    ];
                    $type_name = [
                        'image',
                        'video',
                    ];
                    // Special case for type as we do not want the reference for the
                    // bundle added to the media bundle's referenced entity list.
                    if (is_array($name) && $name[0] == 'referencedEntity') {
                        $values[$langcode][$key] = $referenced_entity->id();
                    } elseif (in_array($field_type, $file_types)) {
                        $asset_token = $entity->getAssetToken();
                        $asset = new Asset($asset_token);
                        $asset->setUri($referenced_entity->getFileUri());
                        $contenthub_entity->addAsset($asset);
                    }
                    // Now add the value, including the "alt" and "title" attributes
                    $data[0] = isset($entity[$key]['alt']) ? $entity[$key]['alt']['title'] : '';
                    $target_url = $entity[$key]['title'];
                    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                } else {
                    $values[$langcode][] = $referenced_entity->uuid();
                }
            } else if (there's nothing in this field, just set it to NULL.
            $values[$langcode] = NULL;
        } else if ($field is a link type) {
            $entity->load($field->validate());
            $entity->normalize($entity);
            // Loop over the items to get the values for each field.
            foreach ($entity as $item) {
                if (isset($item['attributes'])) {
                    $item['unsetItem_attributes'];
                    if (isset($item['key'])) {
                        $key = $item['key'];
                        $value = $item['value'];
                    } else if ($field instanceof PathFieldItemList) {
                        $item['tid'] = '';
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                    } else {
                        $values[$langcode][] = $value;
                    }
                } else if ($attribute = new Attribute($type)) {
                    catch (\Exception $e) {
                        $message = $e->getMessage();
                        throw new ContentHubException($message);
                    }
                    if ($attribute->type == 'array') {
                        $value = array_get($values[$langcode]);
                    } else {
                        $value = array_pop($values[$langcode]);
                    }
                    $attribute->setValue($value, $langcode);
                    // If attribute exists already, append to the existing values.
                    $existing_attribute = $contenthub_entity->getAttribute($name);
                    $attribute->setAttribute($existing_attribute, $attribute->getValues());
                    // Add it to our contenthub entity.
                    $contenthub_entity->addAttribute($name, $attribute);
                    // Allow alterations of the CDF to happen.
                    $entity->entityType->getEntityType();
                    $entity->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
                    // Adds the entity URL to CDF.
                    if (!empty($contenthub_entity->getattribute('url'))) {
                        $entity->getEntityType();
                        $value = file_create_url($entity->getFileUri(), TYPE_STRING);
                        $contenthub_entity->setAttribute('_filepath', $value->setAttribute->setValue($entity->getFileUri()));
                    }
                    default:
                        if ($entity->isViewMode()) {
                            $entity->getLinkTemplate('canonical');
                            $value = $entity->getLinkTemplate();
                            $value->setString();
                            break;
                        }
                        if (isset($value)) {
                            $entity->getAttribute($attribute->type);
                            $entity->getAttribute($attribute->type)->setValue($value, $langcode);
                        }
                    }
                }
            }
        }
    }
    return $contenthub_entity;
}
```



Acquia ContentHub



- 1 method
- 229 lines of code
- 4 separate entity type checks
 - Implies a need for custom logic
- 7 different field name checks
- 2 field instanceof checks

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. $entity->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($excluded_fields as $field) {
        if (!ContentHubEntity::hasField($this) || !ContentHubEntity::hasField($entity)) {
            continue;
        }
        if (!ContentHubEntity::hasField($this->getSerializer()->normalize($field, 'json', $context))) {
            continue;
        }
        $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
        $item = $serialized_field['items'];
        $item['value'] = $entity->getValue();
        $entity->setAttribute('vocabularies', $item);
        $entity->setAttributeValue('vocabulary', $item[0]['target_id'], $langcode);
        continue;
    }
    // To make it work with Paragraphs we are converting the field
    // if ($entity->isOfType('parent') && $entity->getEntityType() === 'paragraph') {
    //     $parent_id = $entity->getEntityTypeId();
    //     $parent_type = $entity->getEntityType();
    //     $parent_value = $entity->getValue();
    //     $parent_type->getStorage($parent_type)->load($parent_id);
    //     $parent_type->setAttribute('parent_id', $langcode);
    //     $contenthub_entity->setAttribute('parent_id', $parent_type);
    // } else {
    //     $contenthub_entity->setAttributeValue('vocabulary', $item[0]['target_id'], $langcode);
    // }
    if ($name == 'bundle' && $entity->getEntityType() === 'media') {
        $contenthub_entity->setAttribute('bundle', $attribute);
        continue;
    }
    // Try to map it to a known field type.
    // Go to the fallback datatype when the field type is not known.
    if ($entity->getEntityType() != $name) {
        $type = $entity->getEntityType();
        elseif ($type == $entity->getEntityType()) {
            $type = $entity->getEntityType();
        }
        if ($type == NULL) {
            $valueField instanceof EntityReferenceFieldItemListInterface) {
                if ($name == $parent_id && $entity->getEntityType() === 'taxonony_term') {
                    $referenced_entities = $storage->loadParents($entity->id());
                } else {
                    $var = \Drupal\core\EntityInterface $referenced_entities;
                }
                $values[$langcode] = [];
                foreach ($referenced_entities as $key => $referenced_entity) {
                    $file_types = [
                        'image',
                        'video',
                    ];
                    $type_name = [
                        'image',
                        'video',
                    ];
                    // Special case for type as we do not want the reference for the
                    // bundle added to the media bundle's referenced entity instance of ConfigEntityBase {
                    if (in_array($name, $file_types)) {
                        $referenced_entity->id();
                    } elseif (in_array($field_type, $file_types)) {
                        $referenced_entity->id();
                    } else {
                        $asset = new Asset($url);
                        $asset->id();
                    }
                    $contenthub_entity->setAttribute('referenced_entity', $referenced_entity);
                    // Now add the value, including the "alt" and "title" attributes
                    $data[] = isset($item[$key]['alt']) ? $item[$key]['alt'] : '';
                    $data[] = isset($item[$key]['title']) ? $item[$key]['title'] : '';
                    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                } else {
                    $values[$langcode][] = $referenced_entity->uid();
                }
            } else {
                if (there's nothing in this field, just set it to NULL.
                $values[$langcode] = NULL;
            } else {
                if ($link_field instanceof LinkFieldHandler) {
                    $item = $link_field->normalizeItem($item);
                }
                // Loop over the items to get the values for each field.
                foreach ($item as $item) {
                    if (isset($item['attributes'])) {
                        $item['unset($item['attributes']);'];
                    }
                    if (isset($item['key'])) {
                        $key = $item['key'];
                        $value = $item['value'];
                    }
                    if ($field instanceof PathFieldItemList) {
                        $item['value'] = '';
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                    }
                    $values[$langcode][] = $value;
                }
            }
        }
        $attribute = new Attribute($type);
        catch (\Exception $e) {
            message('New FormattableMarkup("No type could be registered for type.', $args);
            throw new ContentHubException($message);
        }
        if ($attribute->type() == 'array') {
            $attribute->setAttribute('values');
        } else {
            $value = array_pop($values[$langcode]);
            $attribute->setAttribute($value, $langcode);
        }
        // If attribute exists already, append to the existing values.
        $existing_attribute = $contenthub_entity->getAttribute($name);
        $attribute->setAttribute('existing_attribute', $existing_attribute->getValues());
        // Add it to our contenthub entity.
        $contenthub_entity->setAttribute($name, $attribute);
        // Allow alterations of the CDF to happen.
        $entity->alter();
        $entity->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
        // Adds the entity URL to CDF.
        if (empty($contenthub_entity->getAttribute('url'))) {
            $entity->createUrl($entity->getFileUri(), TYPE_STRING);
            $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
        }
        default:
            if ($entity->isType('new')) {
                $entity->setLinkTemplate('canonical');
                $value = $entity->getLink();
            }
            break;
        if (isset($value)) {
            $entity->setAttribute($attribute, $value);
            $entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
        }
    }
    return $contenthub_entity;
}
```



DrupalCon

SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
 - 229 lines of code
 - 4 separate entity type checks
 - Implies a need for custom logic
 - 7 different field name checks
 - 2 field instanceof checks
 - 3 calls to other local methods

```

ected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    $type_mapping = $this->getFieldTypeMapping($entity);
    // Ignore the entity ID and revision ID.
    Excluded_comes_here = $entity->getExcludedProperties();
    // exclude if this is an excluded field or the current user does not
    // have access to view it
    if (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
        continue;
    }
    // Get the plain version of the field in regular json.
    if ($name == 'metatag') $this->getSerializer()->normalize($field, 'json', $context);
    else {
        $serialized_field = $field->getValue();
        $items = $serialized_field;
        // Given that vocabularies are configuration entities, they are not
        // supported in Content Hub. Instead we use the vocabulary machine name
        // as mechanism to syndicate and import them in the right vocabulary
        if ($name == 'parent_type' && $entity->getEntityType() == 'taxonomy_term') {
            if (!isset($vocabularies[$entity->id()])) {
                $attribute = new Attribute(Attribute::TYPE_STRING);
                $attribute->setValues($items[0]['target_id'], $langcode);
                $contenthub_entity->setAttribute('vocabulary', $attribute);
            } else {
                $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
            }
            continue;
        }
        // To make it work with Paragraphs, we are converting the field
        // entity_id information, because Content Hub cannot deal with
        if ($name == 'parent_uuid' && $entity->getEntityType() == 'paragraph') {
            $parent_type = $entity->parent_type;
            $parent_id = $entity->parent_type->id();
            $parent_value = $entity->parent_type->getValue()['value'];
            $parent_type->getStorage($parent_type)->load($parent_id);
            $attribute = new Attribute($parent_type, $parent_value);
            $contenthub_entity->setAttribute('parent_uuid', $langcode, $attribute);
            continue;
        }
        if ($name == 'bundle' && $entity->getEntityType() == 'media') {
            $attribute = new Attribute(Attribute::TYPE_ARRAY_STRING);
            $attribute->setValue($entity->bundle, $langcode);
            $contenthub_entity->setAttribute('bundle', $attribute);
            continue;
        }
        // Try to map it to a known field type.
        // Go to the fallback datatype when the field type is not known.
        $type = $type_mapping['fallback'];
        if (isset($type_mapping[$name])) {
            $type = $type_mapping[$name];
        } elseif (isset($type_mapping[$field_type])) {
            $type = $type_mapping[$field_type];
        }
        if ($type == NULL) {
            continue;
        }
        $values[$langcode][$field->instanceof EntityReferenceFieldItemListInterface] = [
            'Get_taxonomy_parent_terms' => $entity->getEntityType(),
            'Storage' = $entity->getStorage(),
            'referenced_entities' => $entity->referencedEntities(),
        ];
        foreach ($referenced_entities as $key => $referenced_entity) {
            $file_types = [
                'image',
                'video',
                'type',
                'bundle',
            ];
            // Special case for type as we do not want the reference for the
            // bundle field which stores a media bundle configuration entity UUID.
            if ($entity->name == 'type' && $referenced_entity instanceof ConfigEntityBase) {
                $values[$langcode][$key] = $referenced_entity->id();
            } elseif (in_array($field_type, $file_types)) {
                $uid = $entity->id();
                $asset = file_create_url($referenced_entity->getFileUri());
                $asset->newAsset();
                $asset->setReferencedEntity($uid);
                $contenthub_entity->addAsset($asset);
                // Now add the value, including the "alt" and "title" attributes
                // from the file entity in the field data.
                $data = [
                    'title' => $entity->getItems($key)[0]['alt'] ? $entity->getItems($key)[0]['alt'] : '',
                    'target_uid' => $entity->getItems($key)[0]['title'] ? $entity->getItems($key)[0]['title'] : '',
                    'value' => $entity->getItems($key)[0]['value'],
                ];
                $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
            } else {
                $values[$langcode][] = $referenced_entity->uuid();
            }
        }
    } else if (there's nothing in this field, just set it to NULL.
    $values[$langcode] = NULL;
} else {
    if only if it is a link type
    if ($item = $link_field->normalizeItems($entity)) {
        // Loop over the items to get the values for each field.
        foreach ($item as $item) {
            // Bottfix
            // If we can't find a better solution for this.
            if (isset($item['attributes'])) {
                $keys = is_array($item) ? array_keys($item) : [];
                if (count($keys) > 1) {
                    $value = $item['value'];
                } else {
                    if ($item instanceof PathFieldItemList) {
                        $item = $item->first()->getValue();
                        $item['source'] = '';
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                    }
                }
            }
        }
    }
}
try {
    $attribute = new Attribute($type);
} catch (\Exception $e) {
    $args['type'] = $type;
    $message = 'No type could be registered for %type.', $args;
    throw new ContenthubException($message);
}
if ($stristr($type, 'array')) {
    $value = array_pop($values[$langcode]);
} else {
    $value = array_pop($values[$langcode]);
}
$attribute->setValue($value, $langcode);
// If attribute exists already, append to the existing values.
if ($existing_attribute = $contenthub_entity->getAttribute($name)) {
    $this->appendToAttribute($existing_attribute, $attribute->getValues());
}
// Add it to our contenthub.entity
$contenthub_entity->setAttribute($name, $attribute);

// Allow alterations of the CDF to happen.
context/entity_id = $entity;
$this->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
// Adds the entity URL to CDF.
if (empty($contenthub_entity->getAttribute('url'))) {
    switch ($entity->getEntityType()) {
        case 'file':
            $filepath_attribute = new Attribute($entity->getFilePath(), Attribute::TYPE_STRING);
            $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFilePath()));
            break;
        default:
            // Get entity URL
            if ($entity->isView() && $entity->hasLinkTemplate('canonical')) {
                $uri = $entity->getAbsoluteUrl();
                $value = $uri->toString();
            }
            break;
    }
    if (isset($value)) {
        $url_attribute = new Attribute($entity->getURL(), Attribute::TYPE_STRING);
        $contenthub_entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
    }
}
return $contenthub_entity;

```



Acquia ContentHub



- 1 method
- 229 lines of code
- 4 separate entity type checks
 - Implies a need for custom logic
- 7 different field name checks
- 2 field instanceof checks
- 3 calls to other local methods
- 1 alter hook

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. $entity->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($excluded_fields as $field) {
        if ($entity->access($field->getFieldDefinition()->getName(), $excluded_fields) || !$entity->access('view', $context['account'])) {
            continue;
        }
        // Get the plain version of the field in regular json.
        $serialized_field =& $this->getSerializer()->normalize($field, 'json', $context);
        $item = $serialized_field;
        // Given that vocabulary hub are configuration entities they are not
        // supported in ContentHub. Instead we use the vocabulary machine name
        if ($field->getEntityTypeId() == 'taxonomy_term') {
            if ($contenthub_entity->getAttribute('vocabulary')) {
                $attribute->setValues($entity->target_id, $langcode);
                $contenthub_entity->setAttribute('vocabulary', $attribute);
            } else {
                $contenthub_entity->setAttributeValue('vocabulary', $entity[0]['target_id'], $langcode);
            }
            continue;
        }
        // To make it work with Paragraphs we are converting the field
        // if $parent_id is from parent uuid because Content Hub cannot deal with
        if ($field->getEntityTypeId() == 'paragraph') {
            $parent_id = $entity->getEntityTypeId();
            $parent_type = $entity->getEntityTypeId();
            $parent_value = $entity->getValue()['0']['value'];
            $parent_type->getStorage($parent_type)->load($parent_id);
            $attribute->setValues($parent_id, $langcode);
            $contenthub_entity->setAttribute($parent_id, $attribute);
        }
        continue;
    }
    if ($entity->getEntityType() == 'media') {
        $contenthub_entity->setAttribute('bundle', $attribute);
    }
    // Try to map it to a known field type.
    // Go to the fallback datatype when the field type is not known.
    if ($entity->getEntityType() != 'media') {
        $type = $entity->getEntityType();
        elseif ($type == 'string') {
            $type = $type_mapping[$field_type];
        } elseif ($type == 'file') {
            $type = $type_mapping[$field_type];
        }
        if ($type == NULL) {
            $value_field instanceof EntityReferenceFieldItemListInterface) {
                if ($name = $entity->getEntityTypeId() == 'taxonomy_term') {
                    $referenced_entities = $storage->loadParents($entity->id());
                } else {
                    $var \Drupal\core\EntityInterface $referenced_entities;
                }
                $values[$langcode] = [];
                foreach ($referenced_entities as $key => $referenced_entity) {
                    $file_types = [
                        'image',
                        'video',
                    ];
                    $type_name = [
                        'image',
                        'video',
                    ];
                    // Special case for type as we do not want the reference for the
                    // field added to the media bundle's referenced entity instance of ConfigEntityBase {
                    if (in_array($name, $file_types)) {
                        $referenced_entity->id();
                    } elseif (in_array($field_type, $file_types)) {
                        $referenced_entity->id();
                    } else {
                        $asset = new Asset($url);
                        $asset->setFileUri($referenced_entity->getFileUri());
                        $contenthub_entity->addFile($asset);
                    }
                    // Now add the value, including the "alt" and "title" attributes
                    $data[] = isset($entity[$key]['alt']) ? $entity[$key]['alt'] : '',
                    $data[] = isset($entity[$key]['title']) ? $entity[$key]['title'] : '';
                    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                }
            } else {
                $values[$langcode][] = $referenced_entity->uuid();
            }
        } else {
            if (there's nothing in this field, just set it to NULL.
            $values[$langcode] = NULL;
        } else {
            if ($entity->isLink()) {
                $entity->linkField->normalizeItems($entity);
            }
            // Loop over the items to get the values for each field.
            foreach ($entity->getItems() as $item) {
                if (isset($item->attributes)) {
                    $item->unsetItemAttributes();
                }
                if (empty($item->value)) {
                    $item->value = '';
                }
                if ($field instanceof PathFieldItemList) {
                    $item->value = '';
                    $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                }
                $values[$langcode][] = $value;
            }
        }
        $attribute = new Attribute($type);
        catch (\Exception $e) {
            message('New FormattableMarkup("No type could be registered for type.', $args);
            throw new ContentHubException($message);
        }
        if ($attribute->type == 'array') {
            $attribute->setValues();
        } else {
            $value = array_pop($values[$langcode]);
            $attribute->setValue($value, $langcode);
        }
        // If attribute exists already, append to the existing values.
        $existing_attribute = $contenthub_entity->getAttribute($name);
        $attribute->setAttribute($existing_attribute, $attribute->getValues());
        // Add it to our contenthub entity.
        $contenthub_entity->addAttribute($name, $attribute);
        // Allow alterations of the CDF to happen.
        $entity->alter();
        $entity->alter('alter_acquia_contenthub_cdf', $contenthub_entity, $context);
        // Adds the entity URL to CDF.
        if (!empty($contenthub_entity->getAttribute('url'))) {
            $entity->createUrl($entity->getFileUri(), TYPE_STRING);
            $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
        }
        default:
            if ($entity->getEntityTypeId() == 'entity') {
                $entity->fileCreateUrl($entity->getFileUri(), TYPE_STRING);
                $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
            }
            break;
        if (isset($value)) {
            $entity->setAttribute($attribute, $value);
            $contenthub_entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
        }
    }
    return $contenthub_entity;
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- Why bring up this method? It's not a hook.

Acquia ContentHub



- Why bring up this method? It's not a hook.
- Hooks normalize this sort of coding

Acquia ContentHub



- Why bring up this method? It's not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP

Acquia ContentHub



- Why bring up this method? It's not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP
- Hooks have to concern themselves with implementation details



Acquia ContentHub



- Why bring up this method? It's not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP
- Hooks have to concern themselves with implementation details
- Hooks simultaneously respond to an event and hold logic for all possible returns



DrupalCon
SEATTLE 2019
APRIL 8-12

Drupal 7 module_invoke 'block_view'

```
$array = module_invoke($block->module, 'block_view', $block->delta);
```

```
function system_block_view($delta = '') {
$block = array();
switch ($delta) {
  case 'main':
    $block['subject'] = NULL;
    $block['content'] = drupal_set_page_content();
    return $block;
  case 'powered-by':
    $block['subject'] = NULL;
    $block['content'] = theme('system_powered_by');
    return $block;
  case 'help':
    $block['subject'] = NULL;
    $block['content'] = menu_get_active_help();
    return $block;
  default:
    // All system menu blocks.
    $system_menus = menu_list_system_menus();
    if (isset($system_menus[$delta])) {
      $block['subject'] = t($system_menus[$delta]);
      $block['content'] = menu_tree($delta);
      return $block;
    }
    break;
}
}
```

```
function system_block_view($delta = '') {
$block = array();
switch ($delta) {
  case 'main':
    $block['subject'] = NULL;
    $block['content'] = drupal_set_page_content();
    return $block;
  case 'powered-by':
    $block['subject'] = NULL;
    $block['content'] = theme('system_powered_by');
    return $block;
  case 'help':
    $block['subject'] = NULL;
    $block['content'] = menu_get_active_help();
    return $block;
  default:
    // All system menu blocks.
    $system_menus = menu_list_system_menus();
    if (isset($system_menus[$delta])) {
      $block['subject'] = t($system_menus[$delta]);
      $block['content'] = menu_tree($delta);
      return $block;
    }
    break;
}
}
```

```
function system_block_view($delta = '') {
$block = array();
switch ($delta) {
  case 'main':
    $block['subject'] = NULL;
    $block['content'] = drupal_set_page_content();
    return $block;
  case 'powered-by':
    $block['subject'] = NULL;
    $block['content'] = theme('system_powered_by');
    return $block;
  case 'help':
    $block['subject'] = NULL;
    $block['content'] = menu_get_active_help();
    return $block;
  default:
    // All system menu blocks.
    $system_menus = menu_list_system_menus();
    if (isset($system_menus[$delta])) {
      $block['subject'] = t($system_menus[$delta]);
      $block['content'] = menu_tree($delta);
      return $block;
    }
    break;
}
}
```

```
function system_block_view($delta = '') {
$block = array();
switch ($delta) {
  case 'main':
    $block['subject'] = NULL;
    $block['content'] = drupal_set_page_content();
    return $block;
  case 'powered-by':
    $block['subject'] = NULL;
    $block['content'] = theme('system_powered_by');
    return $block;
  case 'help':
    $block['subject'] = NULL;
    $block['content'] = menu_get_active_help();
    return $block;
  default:
    // All system menu blocks.
    $system_menus = menu_list_system_menus();
    if (isset($system_menus[$delta])) {
      $block['subject'] = t($system_menus[$delta]);
      $block['content'] = menu_tree($delta);
      return $block;
    }
    break;
}
}
```

```
function system_block_view($delta = '') {
$block = array();
switch ($delta) {
  case 'main':
    $block['subject'] = NULL;
    $block['content'] = drupal_set_page_content();
    return $block;
  case 'powered-by':
    $block['subject'] = NULL;
    $block['content'] = theme('system_powered_by');
    return $block;
  case 'help':
    $block['subject'] = NULL;
    $block['content'] = menu_get_active_help();
    return $block;
  default:
    // All system menu blocks.
    $system_menus = menu_list_system_menus();
    if (isset($system_menus[$delta])) {
      $block['subject'] = t($system_menus[$delta]);
      $block['content'] = menu_tree($delta);
      return $block;
    }
    break;
}
}
```



Upgrades and Replacements



“

Perfection is achieved not when there is nothing more to add, but rather when there is nothing more to take away.

”

Antoine de Saint-Exupery



DrupalCon
SEATTLE 2019
APRIL 8-12

Upgrades & Replacements



DrupalCon
SEATTLE 2019
APRIL 8-12

Upgrades & Replacements

■ Plugins



DrupalCon
SEATTLE 2019
APRIL 8-12

Upgrades & Replacements

- Plugins
- Event Subscribers



Spoiler Alert!!



DrupalCon
SEATTLE 2019
APRIL 8-12

Spoiler Alert!!

- This is not a plugins talk



DrupalCon
SEATTLE 2019
APRIL 8-12

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis



DrupalCon
SEATTLE 2019
APRIL 8-12

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
- Plugins usually have a UI interaction.



DrupalCon
SEATTLE 2019
APRIL 8-12

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
- Plugins usually have a UI interaction.
- Plugins are backed by interfaces.

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
- Plugins usually have a UI interaction.
- Plugins are backed by interfaces.
- Plugins are collections of methods that used to be related hook invocations.

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
 - Plugins usually have a UI interaction.
 - Plugins are backed by interfaces.
 - Plugins are collections of methods that used to be related hook invocations.
 - i.e. `hook_block_*`



DrupalCon
SEATTLE 2019
APRIL 8-12

Event Subscribers



DrupalCon
SEATTLE 2019
APRIL 8-12

Event Subscribers

- Individual classes which respond to one or more dispatched events.

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
- Event classes define property mutability

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
 - Event classes define property mutability
 - Can replace both traditional hooks and alter hooks simultaneously

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
 - Event classes define property mutability
 - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
 - Event classes define property mutability
 - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able
- “Propagation” can be stopped

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
 - Event classes define property mutability
 - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able
- “Propagation” can be stopped
- Easily testable



Practical Application



“

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

”

Martin Fowler



Acquia ContentHub



- 1 method
- 229 lines of code
- 4 separate entity type checks
- 7 different field name checks
- 2 field instanceof checks
- 3 calls to other local methods
- 1 alter hook

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
    $fields = $entity->getFields();
    // Get our field mapping. This maps drupal field types to Content Hub
    // type mapping. This is->getFieldTypeMapping($entity);
    // ignore the entity ID and revision ID.
    $excluded_fields = $entity->getExcludedProperties($entity);
    foreach ($excluded_fields as $field) {
        if ($entity->hasField($field) || $entity->isExcludedField($field)) {
            continue;
        }
        // Get the plain version of the field in regular json.
        $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
        $item = $serialized_field;
        // Given that vocabulary hub are configuration entities they are not
        // supported in ContentHub. Instead we use the vocabulary machine name
        if ($field->getEntityTypeId() == 'taxonomy_term') {
            if ($contenthub_entity->getAttribute('vocabulary')) {
                $attribute->setAttributeValue($item['target_id'], $langcode);
                $contenthub_entity->setAttribute('vocabulary', $attribute);
            } else {
                $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
            }
            continue;
        }
        // To make it work with Paragraphs we are converting the field
        // to its parent type because Content Hub cannot deal with
        if ($field->getParentId() && $entity->getEntityType($field->getParentId()) == 'paragraph') {
            $parent_id = $entity->getEntityTypeId();
            $parent_type = $entity->getEntityTypeId();
            $parent_value = $entity->getValue()['value'];
            $parent_type->getStorage($parent_type)->load($parent_id);
            $attribute->setValue($parent_id, $langcode);
            $contenthub_entity->setAttribute('parent_id', $attribute);
            continue;
        }
        if ($name == 'bundle' && $entity->getEntityType() == 'media') {
            $contenthub_entity->setAttribute('bundle', $attribute);
            continue;
        }
        // Try to map it to a known field type.
        // Go to the fallback datatype when the field type is not known.
        if ($entity->getEntityType($name)) {
            $type = $entity->getEntityType($name);
            elseif ($type->getMappingType($field_type)) {
                $type = $type->getMappingType($field_type);
            }
            if ($type == NULL) {
                $value_field = $entity->getInstanceof(EntityReferenceFieldItemListInterface);
                if ($name == 'parent' && $entity->getEntityType() == 'taxonomy_term') {
                    $referenced_entities = $storage->loadParents($entity->id());
                    $entity->referenced_entities->referenced_entities();
                }
                $values[$langcode] = [];
                foreach ($referenced_entities as $key => $referenced_entity) {
                    $file_types = [
                        'image',
                        'video',
                    ];
                    $type_name = [
                        'image',
                        'video',
                    ];
                    // Special case for type as we do not want the reference for the
                    // bundle added to the media bundle's referenced entity instance of ConfigEntityBase {
                    if (in_array($name, $file_types) && $referenced_entity->id() != $entity->id()) {
                        $entity->referenced_entities->referenced_entities->add($referenced_entity->id());
                    }
                    elseif (in_array($field_type, $file_types)) {
                        $entity->referenced_entities->referenced_entities->add($referenced_entity->id());
                    }
                    elseif ($entity->referenced_entities->referenced_entities->id() == $referenced_entity->id()) {
                        $entity->referenced_entities->referenced_entities->add($referenced_entity->id());
                    }
                    // Now add the value, including the "alt" and "title" attributes
                    $data[] = [
                        'set' => $entity[$key]['alt'],
                        'target' => $entity[$key]['title'],
                    ];
                    $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                }
                else {
                    $values[$langcode][] = $referenced_entity->uuid();
                }
            }
            else {
                if (there's nothing in this field, just set it to NULL.
                $values[$langcode] = NULL;
            }
            else {
                if ($entity->isLinkField($field)) {
                    $entity->linkField->normalizeItems($entity);
                }
                // Loop over the items to get the values for each field.
                foreach ($entity->items as $item) {
                    if ($entity->isSet($item['attributes'])) {
                        $entity->unset($item['attributes']);
                    }
                    if ($entity->isKey($item['key']) && $entity->isSet($item['value'])) {
                        $entity->value[$item['key']] = $item['value'];
                    }
                    if ($field->instanceof(PathFieldItemList)) {
                        $item['filepath'] = '';
                        $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
                        $values[$langcode][] = $value;
                    }
                    try {
                        $attribute = new Attribute($type);
                    }
                    catch (\Exception $e) {
                        $message = 'No type could be registered for type.';
                        throw new ContentHubException($message);
                    }
                    if ($attribute->type == 'array') {
                        $value = array_pop($values[$langcode]);
                        $attribute->setValue($value, $langcode);
                    }
                    else {
                        $value = array_pop($values[$langcode]);
                        $attribute->setValue($value, $langcode);
                    }
                    if ($attribute->exists($value)) {
                        $entity->setAttribute($name, $attribute, $entity->getValues());
                    }
                    else {
                        $entity->setAttribute($name, $attribute);
                    }
                    $entity->addAttribute($name, $attribute);
                }
                // Add it to our contenthub entity.
                $contenthub_entity->setAttribute($name, $attribute);
                // Allow alterations of the CDF to happen.
                $entity->entityType->entityTypeHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);
                // Adds the entity URL to CDF.
                if (!empty($contenthub_entity->getAttribute('url'))) {
                    $entity->entityType->createMarkup($entity->getFileUri(), TYPE_STRING);
                    $contenthub_entity->setAttribute('_filepath', $filepath->setAttribute->setValue($entity->getFileUri()));
                }
                default:
                    if ($entity->isKey($value)) {
                        $entity->setKey($value);
                    }
                    if ($entity->isLinkTemplate('canonical')) {
                        $value = $entity->getLinkTemplate();
                    }
                    break;
                if ($entity->isAttribute($value)) {
                    $entity->setAttribute($name, $value->setAttribute->setValue($value, $langcode));
                }
            }
        }
    }
    return $contenthub_entity;
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



■ 1 method

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub



- 1 method
- 58 lines of code

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {  
    if (!$cdf->hasEntities()) {  
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");  
    }  
    $event = new PruneCdfEntitiesEvent($cdf);  
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);  
    $cdf = $event->getCdf();  
    $this->handleModules($cdf, $stack);  
    // Allows entity data to be manipulated before unserialization.  
    $event = new EntityDataTamperEvent($cdf, $stack);  
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);  
    $cdf = $event->getCdf();  
    // Organize the entities into a dependency chain.  
    // Use a while loop to prevent memory expansion due to recursion.  
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {  
        // @todo add tracking to break out of the while loop when dependencies cannot  
        be further processed.  
        $count = count($stack->getDependencies());  
        foreach ($cdf->getEntities() as $uuid => $entity_data) {  
            if (!$stack->hasDependency($uuid) && $this->  
                entityIsProcessable($entity_data, $stack)) {  
                $event = new LoadLocalEntityEvent($entity_data, $stack);  
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,  
                $event);  
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event->  
                    getEntity());  
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);  
                $entity = $event->getEntity();  
                if ($entity) {  
                    $entity->save();  
                    $wrapper = new DependentEntityWrapper($entity);  
                    // Config uuids can be more fluid since they can match on id.  
                    if ($wrapper->getUuid() != $uuid) {  
                        $wrapper->setRemoteUuid($uuid);  
                    }  
                    $stack->addDependency($wrapper);  
                    if ($entity->isNew()) {  
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;  
                        $event = new EntityImportEvent($entity, $entity_data);  
                    }  
                    else {  
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;  
                        $event = new EntityImportEvent($entity, $entity_data);  
                    }  
                    $this->dispatcher->dispatch($event_name, $event);  
                }  
                else {  
                    // Remove CDF Entities that were processable but didn't resolve into  
                    // an entity.  
                    $cdf->removeCDFEntity($uuid);  
                }  
            }  
        }  
        if ($count === count($stack->getDependencies())) {  
            // @todo get import failure logging and tracking working.  
            $event = new FailedImportEvent($cdf, $stack, $count);  
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);  
            if ($event->hasException()) {  
                throw $event->getException();  
            }  
        }  
    }  
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        // be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        // be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method
- 0 alter hook

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        // be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



Acquia ContentHub



- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method
- 0 alter hook
- 5 dispatched events

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
    if (!$cdf->hasEntities()) {
        throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
    }
    $event = new PruneCdfEntitiesEvent($cdf);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
    $cdf = $event->getCdf();
    $this->handleModules($cdf, $stack);
    // Allows entity data to be manipulated before unserialization.
    $event = new EntityDataTamperEvent($cdf, $stack);
    $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
    $cdf = $event->getCdf();
    // Organize the entities into a dependency chain.
    // Use a while loop to prevent memory expansion due to recursion.
    while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
        // @todo add tracking to break out of the while loop when dependencies cannot
        // be further processed.
        $count = count($stack->getDependencies());
        foreach ($cdf->getEntities() as $uuid => $entity_data) {
            if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
                $event = new LoadLocalEntityEvent($entity_data, $stack);
                $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
                $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
                $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
                $entity = $event->getEntity();
                if ($entity) {
                    $entity->save();
                    $wrapper = new DependentEntityWrapper($entity);
                    // Config uuids can be more fluid since they can match on id.
                    if ($wrapper->getUuid() != $uuid) {
                        $wrapper->setRemoteUuid($uuid);
                    }
                    $stack->addDependency($wrapper);
                    if ($entity->isNew()) {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
                        $event = new EntityImportEvent($entity, $entity_data);
                    } else {
                        $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
                        $event = new EntityImportEvent($entity, $entity_data);
                    }
                    $this->dispatcher->dispatch($event_name, $event);
                } else {
                    // Remove CDF Entities that were processable but didn't resolve into
                    // an entity.
                    $cdf->removeCDFEntity($uuid);
                }
            }
        }
        if ($count === count($stack->getDependencies())) {
            // @todo get import failure logging and tracking working.
            $event = new FailedImportEvent($cdf, $stack, $count);
            $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
            if ($event->hasException()) {
                throw $event->getException();
            }
        }
    }
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub 8.x-2.x

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

Acquia ContentHub 8.x-2.x

```
public static function getSubscribedEvents() {
  $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY] [] = ['onLoadLocalEntity', 10];
  return $events;
}

public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
  [CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id,
  $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
}
```



Acquia ContentHub 8.x-2.x

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
$cdf = $event->getCdf();
if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
}
$attribute = $cdf->getAttribute('entity_type');
// We only care about user entities.
if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
}
// Don't do anything with anonymous users.
if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
}
$mail_attribute = $cdf->getAttribute('mail');
if (!$mail_attribute) {
    return;
}
$mail = $mail_attribute->getValue()['und'];
/** @var \Drupal\user\UserInterface $account */
$account = user_load_by_mail($mail);
if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
}
```



Pseudo-hook-comparison

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }
    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```



Pseudo-hook-comparison

```
public function onUploadLocalEntity(UploadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }

    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```



Pseudo-hook-comparison

```
public function onUploadLocalEntity(UploadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }

    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

Event Definition

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```



Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id,
$cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }
}
```



Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id,
$cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }
}
```



Event Definition

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf():\Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack():\Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity():\Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Definition

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getcdf() \Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack():\Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity():\Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Definition

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf() \Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack() \Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity():\Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Definition

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf() \Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack() \Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity() \Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Definition

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf() \Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack() \Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity() \Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface) void



Event Propagation

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf():\Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack():\Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation():void
- m 🔑 getEntity():\Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Propagation

c LoadLocalEntityEvent

- f ♀ cdf:\Acquia\ContentHubClient\CDF\CDFObject
- f ♀ stack:\Drupal\depcalc\DependencyStack
- f ♀ entity:\Drupal\Core\Entity\EntityInterface
- f 🔒 propagationStopped:bool
- m 🔑 __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
- m 🔑 isPropagationStopped():bool
- m 🔑 getCdf():\Acquia\ContentHubClient\CDF\CDFObject
- m 🔑 getStack():\Drupal\depcalc\DependencyStack
- m 🔑 stopPropagation() void
- m 🔑 getEntity():\Drupal\Core\Entity\EntityInterface
- m 🔑 hasEntity():bool
- m 🔑 setEntity(entity : \Drupal\Core\Entity\EntityInterface):void



Event Propagation

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }
    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```



Event Propagation

```
public function onLoadLocalEntity(OnLoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }

    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```



Event Propagation

```
public function onLoadLocalEntity(OnLoadLocalEntityEvent $event) {
    $cdf = $event->getCdf();
    $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
    if ($entity = $this->getRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
        $event->setEntity($entity);
        $event->stopPropagation();
    }

    $cdf = $event->getCdf();
    if ($cdf->getType() !== 'drupal8_content_entity') {
        return;
    }
    $attribute = $cdf->getAttribute('entity_type');
    // We only care about user entities.
    if (!$attribute || $attribute->getValue()['und'] !== 'user') {
        return;
    }
    // Don't do anything with anonymous users.
    if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
        return;
    }
    $mail_attribute = $cdf->getAttribute('mail');
    if (!$mail_attribute) {
        return;
    }
    $mail = $mail_attribute->getValue()['und'];
    /** @var \Drupal\user\UserInterface $account */
    $account = user_load_by_mail($mail);
    if ($account) {
        $event->setEntity($account);
        $event->stopPropagation();
        return;
    }
}
```





DrupalCon
SEATTLE 2019
APRIL 8-12

“Event Propagation” in D7



DrupalCon
SEATTLE 2019
APRIL 8-12

“Event Propagation” in D7

- module_implements()



“Event Propagation” in D7

- `module_implements()`
- `module_invoke()`

“Event Propagation” in D7

- `module_implements()`
 - `module_invoke()`
- `module_invoke_all()`



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Priority



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Priority

- EventSubscribers default to a 0 priority



EventSubscriber Priority

- EventSubscribers default to a 0 priority
- Priorities of same weight non-deterministically execute
- Priority is opposite of weight (higher priority happens earlier)



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Priority

```
public static function getSubscribedEvents() {  
    $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY] [] = ['onLoadLocalEntity', 10];  
    return $events;  
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Priority

```
public static function getSubscribedEvents() {
  $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY] [] = ['onLoadLocalEntity', 10];
  return $events;
}
```



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Testing



DrupalCon
SEATTLE 2019
APRIL 8-12

EventSubscriber Testing

- Mock the dependencies

EventSubscriber Testing

- Mock the dependencies
- Create the event

EventSubscriber Testing

- Mock the dependencies
- Create the event
- Pass it to the public method on the event subscriber

EventSubscriber Testing

- Mock the dependencies
- Create the event
- Pass it to the public method on the event subscriber
- Assert as necessary



DrupalCon
SEATTLE 2019
APRIL 8-12

Key Take Aways



DrupalCon
SEATTLE 2019
APRIL 8-12

Key Take Aways

- Drupal hooks are a solution for a bygone era



Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer



Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer
- We should adopt other available solutions in core today



Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer
- We should adopt other available solutions in core today
- Profit



Questions & Answers



DrupalCon
SEATTLE 2019
APRIL 8-12

Join us for contribution opportunities

Friday, April 12, 2019

Mentored Contributions

9:00-18:00
Room: 602

First Time Contributor Workshop

9:00-12:00
Room: 606

General Contributions

9:00-18:00
Room: 6A

#DrupalContributions



DrupalCon
SEATTLE 2019
APRIL 8-12

What did you think?

Locate this session at the DrupalCon Seattle website:

<http://seattle2019.drupal.org/schedule>

Take the Survey!

<https://www.surveymonkey.com/r/DrupalConSeattle>

Thank you!