



amaze.io  
part of Mirantis

# No-code API Integrations with ECA and HTTP Client Manager

**DrupalCamp DEN**

January 2023



**Brandon Williams**  
Software Engineer

Hello, my name is Brandon Williams and I'm a software engineer with amaze.io. Today I'm excited to share with you a way to integrate APIs into your Drupal sites without having to write any PHP code. In this short presentation I'll give you an overview of the ECA and HTTP Client Manager modules, how they are typically used, and some examples of the how they can be combined.

---

## ECA: Event - Condition - Action

The no-code solution that empowers you to orchestrate your Drupal site. It's a powerful, versatile, and user-friendly **rules engine** for Drupal 9+.

<https://dgo.re/eca>

Stable - Works with Drupal ^9.4 || ^10

First on the list is the ECA: event, condition, action module. It's described as "the no-code solution that empowers you to orchestrate your Drupal site. It's a powerful, versatile, and user-friendly rules engine for Drupal 9+." It has a stable release which is covered by the security team and is actively being developed, having gone from its' first commit to a stable release in just one year.

---

## Overview

- Replacement for Rules in Drupal 9+
- Reduces the need for custom modules
- Can replace contrib helper modules
  - Redirect after login
  - Computed field
  - Etc
- Separate processor and UI modules
- Requires only Drupal core
- Fast and searchable docs with examples library <https://ecaguide.org/>

The most prominent selling point is that it's a replacement for the rules module, which still doesn't have a stable version. But it's much more than just a drop-in replacement for rules. ECA is able to take advantage of a much wider set of event and action plugins in Drupal, allowing you to do things in the UI that would've required custom modules before. Things like form alters and custom cron tasks can now be done entirely in ECA.

It can also replace some helper modules from contrib like redirect after login, computed fields, and much more. This can lower the number of contrib modules being used which makes it easier to keep track of upgrades. I can also imagine a scenario where a contrib module still doesn't have a stable release, but your client or organization requires stable versions in production, so you could potentially replace it with an ECA model instead.

ECA itself is very lean, it only requires Drupal core and a modeller plugin to get started. The project page on drupal.org is very detailed and the maintainers have created a separate documentation site at [ecaguide.org](https://ecaguide.org/) which is fast and searchable.

---

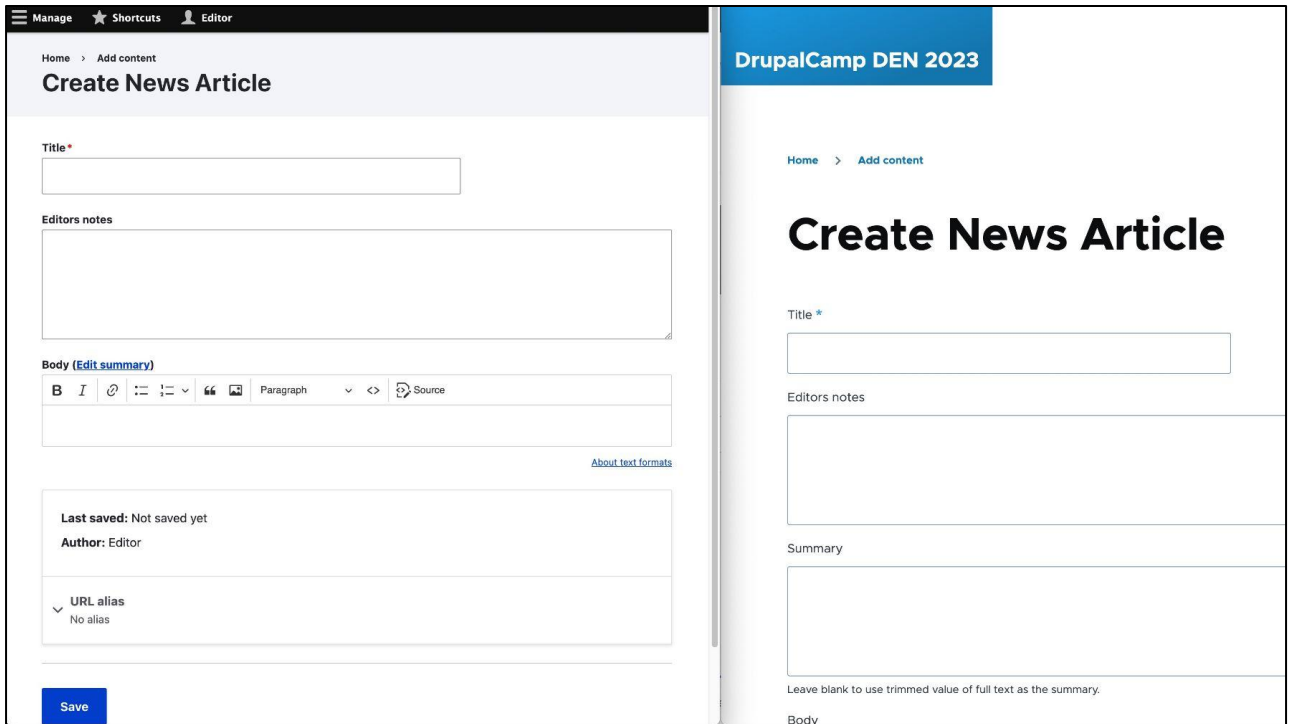
## Sub-modules

Plugins are organized into sub-modules so you can enable only what you need.

- ECA Access
- ECA Base
- ECA Cache
- ECA Config
- ECA Content
- ECA Endpoint
- ECA Form
- ECA Log
- ECA Migrate
- ECA Misc
- ECA Queue
- ECA Render
- ECA User
- ECA Views
- ECA Workflow

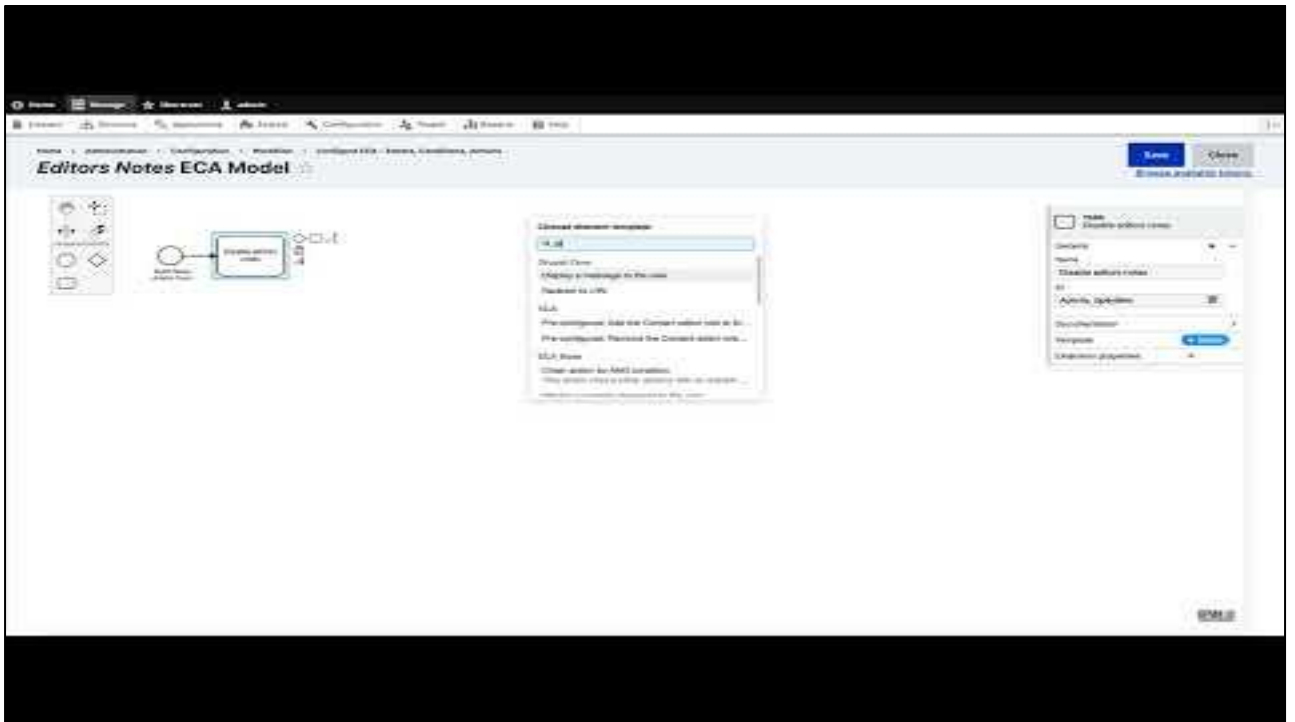
ECA has several sub modules to organize related events, conditions, and actions. For example, the ECA User sub module has events for “login of a user” and “cancelling of a user.” ECA Form has actions for “Form field: set as required” and “Form state: set redirect.” And the rest of the sub modules pretty self explanatory. All of the available plugins are documented on [ecaguide.org](http://ecaguide.org).

So far everything I could think of was already covered and one set of actions that didn't exist yet had a patch available within 24 hours of me asking about it.



Let's look at a small example of what ECA can do. Here I have a content type called News Article with a text field called "editors notes." On the left I'm logged in as an editor and on the right I'm logged in as a regular user. Let's say I have a requirement that only editors are allowed to make changes to the editors notes fields. You could use the `field_permissions` module to do this, but for most of Drupal 8 it didn't have a stable release so the only option was to create a custom module with a `hook_entity_field_access` plugin.

Instead I'll create an ECA model to do this.

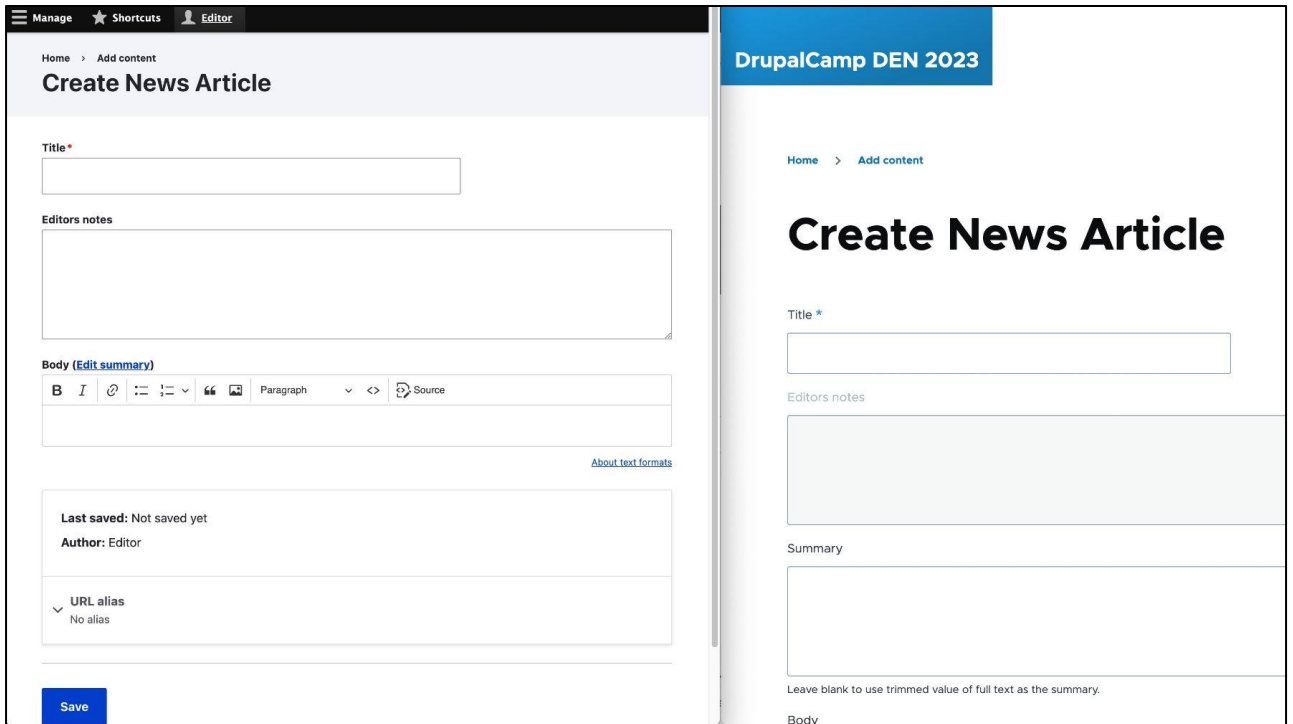


\* Click image to play video \*

I've created a new ECA model and first thing I'm going to do is to give it a name of "Editors Notes" and save. Then I'll add a new event and in the templates section I'll search for the build form event. I name it "Build News Article Form" to be more descriptive and set the properties so that the event only applies to nodes of type "news article."

Then I add an action after the event and I know I want it to disable the editors notes field. In the templates section I search for "disable" and find an action for "Form field: set as disabled" where I can give the machine name of the field to disable.

Last thing I want to do is make sure the field isn't disabled for content editors so I can add a condition on the action for the users current role, select the Content Editor and negate the condition and that's the whole model done.



With our new ECA model in place you can see that the “editors notes” field is disabled for the regular user on the right but not for the editor on the left.

---

## HTTP Client Manager

Allows you to manage HTTP clients in a simple and efficient way by describing web service APIs with YAML, JSON or PHP files.

[https://dgo.re/http\\_client\\_manager](https://dgo.re/http_client_manager)

Stable - Works with Drupal ^9.4 || ^10

The next module is HTTP Client manager. It allows you to manage HTTP clients in a simple and efficient way by describing web service APIs with YAML, JSON or PHP files. It has a stable release which is covered by the security team and has versions compatible with Drupal 8 through 10.



# SWAPI

The Star Wars API  
(what happened to swapi.co?)

Try it now!

<https://swapi.dev/api/>

Need a hint? try [people/1/](#) or [planets/3/](#) or [starships/8/](#)

Result:

```
{
  "name": "Luke Skywalker",
  "height": "172",
  "mass": "77",
  "hair_color": "blond",
  "skin_color": "fair",
  "eye_color": "blue",
  "birth_year": "1989",
  "gender": "male",
  "homeworld": "https://swapi.dev/api/planets/1/",
  "films": [
    "https://swapi.dev/api/films/2/",
    "https://swapi.dev/api/films/6/",
    "https://swapi.dev/api/films/3/",
    "https://swapi.dev/api/films/1/",
    "https://swapi.dev/api/films/7/"
  ],
  "species": [
```

What is this?

The Star Wars API, or "swapi" (Swah-pee) is the world's first quantified and programmatically-accessible data source for all the data from the Star Wars canon universe!

We've taken all the rich contextual stuff from the universe and

How can I use it?

All the data is accessible through our HTTP web API. Consult our [documentation](#) if you'd like to get started.

Helper libraries for popular programming languages are also provided so you can consume swapi in your favourite programming language, in a style that suits you.

What happened with old swapi.co?

swapi.co is not supported and maintained anymore. But since so many projects and tutorials used it as their educational playground, this is an "unofficial" branch.

This project is open source and you can contribute on [GitHub](#).

I think the best way to explain this module is to show you an example implementation. I decided to describe the Star Wars API, which is available at S W A P I dot dev. It's a free and open REST API that allows anyone to get information about the star wars films. You can query for information about planets, spaceships, vehicles, people, films, and species.

In this screenshot I'm showing you how querying the API for the person with ID "one" will return information about Luke Skywalker, including his height, eye color, and other properties.

```
! swapi.yml ×
1 name: Star Wars API
2 description: Get information about the Star Wars films
3 imports:
4   - resources/person.yml
5 operations:
6   PersonByID:
7     summary: Get a specific person by ID
8     httpMethod: GET
9     uri: people/{personId}
10    responseModel: Person
11    parameters:
12      personId:
13        type: string
14        location: uri
15        description: Star Wars Person ID
16        required: true
17

! person.yml ×
1 models:
2   Person:
3     type: object
4     properties:
5       name:
6         location: json
7         type: string
8 > height: ...
11 > mass: ...
14 > hair_color: ...
17 > skin_color: ...
20 > eye_color: ...
23 > birth_year: ...
26 > gender: ...
29 > created: ...
32 > edited: ...
35
```

To make this API data available to Drupal I'll create a new client for it using the HTTP client manager. Each API client gets its own definition and for this example I picked YAML as the language to use. I'm only showing you a basic version for now, but I'll provide a link to a full example later on.

In addition to some meta information like the client name and description, the module needs to know what request operations can be used and how the responses will be structured.

```
! swapi.yml ×
1 name: Star Wars API
2 description: Get information about the Star Wars films
3 imports:
4   - resources/person.yml
5 operations:
6   PersonByID:
7     summary: Get a specific person by ID
8     httpMethod: GET
9     uri: people/{personId}
10    responseModel: Person
11    parameters:
12      personId:
13        type: string
14        location: uri
15        description: Star Wars Person ID
16        required: true
17

! person.yml ×
1 models:
2   Person:
3     type: object
4     properties:
5       name:
6         location: json
7         type: string
8 > height: ...
11 > mass: ...
14 > hair_color: ...
17 > skin_color: ...
20 > eye_color: ...
23 > birth_year: ...
26 > gender: ...
29 > created: ...
32 > edited: ...
35
```

Here I've defined one request operation called PersonByID. I've configured it to use a GET request, use the "person" url, and required a single parameter called personID.

```
! swapi.yml ×
1 name: Star Wars API
2 description: Get information about the Star Wars films
3 imports:
4   - resources/person.yml
5 operations:
6   PersonByID:
7     summary: Get a specific person by ID
8     httpMethod: GET
9     uri: people/{personId}
10    responseModel: Person
11    parameters:
12      personId:
13        type: string
14        location: uri
15        description: Star Wars Person ID
16        required: true
17

! person.yml ×
1 models:
2   Person:
3     type: object
4     properties:
5       name:
6         location: json
7         type: string
8       height: ...
9       mass: ...
10      hair_color: ...
11      skin_color: ...
12      eye_color: ...
13      birth_year: ...
14      gender: ...
15      created: ...
16      edited: ...
17
```

In order for the HTTP client to return data, you must configure a model with the expected response structure and link the model to the operation. Here I have a Person model with ten properties I know will be returned from the API as strings.

The screenshot shows the Drupal administration interface for the HTTP Client Manager module. The breadcrumb trail is: Home > Administration > Configuration > Web services. The page title is "HTTP Client Manager" with a star icon. Below the title, there are two tabs: "HTTP Services API" (which is active) and "Settings".

ID	Title	Parent	Base URI	Operations
swapi	Star Wars API		https://swapi.dev/api/	<a href="#">View Commands</a>

After saving the client definition, it's able to be used in Drupal. The HTTP client manager includes an admin UI to list all the available clients and their commands. This page lists all the available APIs that are configured with the module. Clicking "view commands" will give you a list of commands that can be used from that api client.

Back to site | Manage | Shortcuts | admin

Content | Structure | Appearance | Extend | Configuration | People | Reports | Help

Home > Administration > Configuration > Web services > HTTP Client Manager

## Star Wars API ☆

- All Commands - ▾

- FilmByID
- Films
- People
- PersonByID**
- PlanetByID
- Planets
- SpaceshipByID
- Spaceships
- Species
- SpeciesByID
- VehicleByID
- Vehicles

Get a specific person by ID

HTTP Method	URI	Operations
GET	people/{personId}	Configured Requests (1)

Parameters

Name	Type	Default	Description	Required	Location
personId	String		Star Wars Person ID	Yes	uri

Here is the list of all commands for the star wars api client. I've selected the PersonByID operation that was defined in YAML. You can see the same information about the request like the GET method, "people" url, and required "personID" parameter.

Since most API operations frequently require passing parameters to query the desired information, the module allows you to make "configured requests" which are basically a way to "save" those parameters.

The screenshot shows a web application interface with a top navigation bar containing 'Back to site', 'Manage', 'Shortcuts', and 'admin'. Below this is a secondary navigation bar with 'Content', 'Structure', 'Appearance', 'Extend', 'Configuration', 'People', 'Reports', and 'Help'. The main content area has a breadcrumb trail: 'Home > Administration > Configuration > Web services > HTTP Client Manager > Star Wars API'. The title is 'Http Config Request' with a star icon. A blue button '+ Add Http Config Request' is located above a table. The table has four columns: 'Http Config Request', 'Machine name', 'Parameters', and 'Operations'. One row is visible with the following data: 'Luke Skywalker', 'luke\_skywalker', and '• personId: 1'. The 'Operations' column for this row contains a dropdown menu with 'Edit', 'Delete', and 'Execute' options.

Http Config Request	Machine name	Parameters	Operations
Luke Skywalker	luke_skywalker	• personId: 1	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Execute</a>

Our PersonByID operation requires a personID parameter, so here I've created a new Luke Skywalker request where the personID is hard coded to one. Clicking execute will query the API with the saved parameter.

Back to site | Manage | Shortcuts | admin

Content | Structure | Appearance | Extend | Configuration | People | Reports | Help

Home > Administration > Configuration > Web services > HTTP Client Manager > Star Wars API > Http Config Request

## Http Config Request execution ☆

```
REQUEST: luke_skywalker

Array
(
  [serviceApi] => swapi
  [commandName] => PersonByID
  [parameters] => Array
    (
      [personId] => 1
    )
)
```

```
RESPONSE: luke_skywalker

GuzzleHttp\Command\Result Object
(
  [name] => Luke Skywalker
  [height] => 172
  [mass] => 77
  [hair_color] => blond
  [skin_color] => fair
  [eye_color] => blue
  [birth_year] => 19BBY
  [gender] => male
  [created] => 2014-12-09T13:50:51.644000Z
  [edited] => 2014-12-20T21:17:56.891000Z
)
```

And here is the result. You can see that Drupal was able to query the star wars API about person “one” and return information about Luke Skywalker. This part of the admin UI is just a convenient way to test the “configured requests,” it doesn’t actually make use of the data.



- 
- HTTP Client usage in code

```
$client = \Drupal::service('http_client_manager.factory')->get('swapi');  
$luke = $client->personById(['personId' => 1]);
```

- No-code usage with ECA

The main purpose of the HTTP client manager module is to take your API definitions in YAML and convert them into clients that can be used in PHP. So here is a small example of how a developer can load the star wars api from the Drupal service factory and call the “PersonByID” operation to get information about Luke Skywalker.

A recent addition to the module added actions support for all api clients and configured requests which means they are automatically available for use with ECA. So now you can use the defined api clients as ECA actions without writing any PHP code.

Let's take a look at a few examples.

---

## ECA + HTTP Client Manager

Create a content type for a Star Wars “person” that automatically fills in data from the Star Wars API

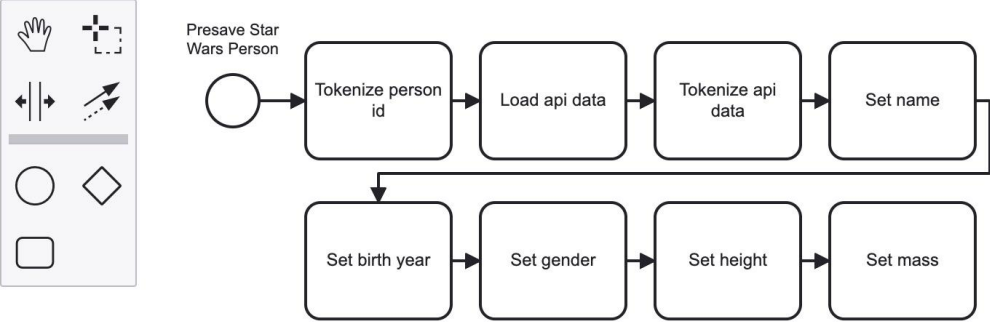
The first example I have will utilize the star wars api that was just defined. The scenario I’ve created is that our Drupal site has a “person” content type, and when a user creates content of that type they are asked to select a person from the star wars universe. Then when they hit save, Drupal should pull all the information about that person and save it on the node.

Back to site Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

## Fill in SWAPI Person Data ECA Model ☆



The diagram illustrates an ECA model for filling in SWAPI person data. It starts with a start node (circle) labeled "Presave Star Wars Person". The flow proceeds through a sequence of actions: "Tokenize person id", "Load api data", "Tokenize api data", and "Set name". From the "Set name" action, the flow branches into two parallel paths. The first path continues with "Set birth year", "Set gender", "Set height", and "Set mass". The second path is a direct connection from "Set name" to "Set mass". A toolbar on the left contains icons for hand, zoom, pan, and other workflow editing tools.

```
graph LR; Start((Presave Star Wars Person)) --> TokenizePerson[Tokenize person id]; TokenizePerson --> LoadAPI[Load api data]; LoadAPI --> TokenizeAPI[Tokenize api data]; TokenizeAPI --> SetName[Set name]; SetName --> SetBirth[Set birth year]; SetName --> SetMass[Set mass]; SetBirth --> SetGender[Set gender]; SetGender --> SetHeight[Set height]; SetHeight --> SetMass;
```

I already showed you how I defined the star wars API in the HTTP client manager module so now I'll show you what the ECA model looks like.

The screenshot shows a BPMN configuration interface for a process titled "Fill in SWAPI Person Data ECA Model". The interface includes a top navigation bar with "Home", "Manage", "Shortcuts", and "admin". Below this is a secondary navigation bar with "Content", "Structure", "Appearance", "Extend", "Configuration", "People", "Reports", and "Help". The main content area displays a BPMN diagram with the following elements:

- A "Presave Star Wars Person" event (circle with a lightning bolt) is connected to a "Load api data" task (rounded rectangle).
- The "Load api data" task is connected to a "Tokenize api data" task.
- The "Tokenize api data" task is connected to a "Set name" task.
- The "Set name" task is connected to a "Set birth year" task.
- The "Set birth year" task is connected to a "Set gender" task.
- The "Set gender" task is connected to a "Set height" task.
- The "Set height" task is connected to a "Set mass" task.

On the right side of the interface, there is a configuration panel for the "PRESAVE CONTENT ENTITY" event. The panel includes the following settings:

- General: Presave Star Wars Person
- Template: Applied
- Custom properties: Type (and bundle) dropdown menu with "Content: Star Wars API Person" selected.

The BPMN logo is visible in the bottom right corner of the interface.

I added a “presave content entity” event for set it to fire only for “star wars person” content types. The presave event will be called for new and updated entities.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Fill in SWAPI Person Data ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; Start((Pressure Star Wars Person)) --> TokenizeID[Tokenize person id]; TokenizeID --> LoadAPI[Load api data]; LoadAPI --> TokenizeAPI[Tokenize api data]; TokenizeAPI --> SetName[Set name]; SetName --> SetBirth[Set birth year]; SetBirth --> SetGender[Set gender]; SetGender --> SetHeight[Set height]; SetHeight --> SetMass[Set mass];
```

ENTITY: GET FIELD VALUE  
Tokenize person id

General

Template Applied

Custom properties

Field name

field\_swapi\_person\_search  
The machine name of the field, that holds the value. This property supports tokens.

Name of token

pid  
The field value will be loaded into this specified token.

Entity

Provide the token name of the entity that this action should operate with.

BPMN.IO

I need a person ID to query so I add a “entity: get field value” action and give it the machine name of the field and the token name it should use.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Fill in SWAPI Person Data ECA Model ☆

Save Close [Browse available tokens.](#)

The diagram shows a BPMN process for 'Fill in SWAPI Person Data ECA Model'. It starts with a start event 'Provide Star Wars Person'. The process flow is as follows: 'Tokenize person id' (task) -> 'Load api data' (task, highlighted with a blue border) -> 'Tokenize api data' (task) -> 'Set name' (task). From 'Set name', the flow goes to 'Set birth year' (task) -> 'Set gender' (task) -> 'Set height' (task) -> 'Set mass' (task). A toolbar on the left contains icons for zooming, deleting, and adding elements. A configuration panel on the right is open for the 'Load api data' task.

**STAR WARS API - GET A SPECIFIC ...**  
Load api data

General >

Template **Applied** >

Custom properties >

Star Wars Person ID

[pid]

Replace tokens

yes

When enabled, tokens will be replaced before executing the action. Please note: Actions might already take care of replacing tokens on their own. Therefore, use this option only with care and when it makes sense.

BPMN.IO

The next action is a call to the star wars api via the HTTP client and I pass it the pid token that was made in the previous action.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Fill in SWAPI Person Data ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; Start((Pressive Star Wars Person)) --> TokenizePerson[Tokenize person id]; TokenizePerson --> LoadAPI[Load api data]; LoadAPI --> TokenizeAPI[Tokenize api data]; TokenizeAPI --> SetName[Set name]; SetName --> MergeGateway{ }; MergeGateway --> SetBirthYear[Set birth year]; MergeGateway --> SetGender[Set gender]; MergeGateway --> SetHeight[Set height]; MergeGateway --> SetMass[Set mass];
```

PRIVATE TEMPORARY STORE: RE...  
Tokenize api data

General >

Template Applied >

Custom properties >

Collection

http\_client\_manager  
The collection of the store.

Store key

last result  
The key of the value in the store.

Name of token

api\_person  
The name of the token, the value is stored into.

BPMN.IO

The client call will save the api results to a private temporary storage so this action retrieves that data and stores it in a token named “api\_person”.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Fill in SWAPI Person Data ECA Model ☆

Save Close [Browse available tokens.](#)

The workflow diagram shows a sequence of actions: 'Pressive Star Wars Person' (start), 'Tokenize person id', 'Load api data', 'Tokenize api data', 'Set name', 'Set birth year', 'Set gender', 'Set height', and 'Set mass'. The 'Set name' action is highlighted with a blue border.

**ENTITY: SET FIELD VALUE**  
Set name

General

Template **Applied**

Custom properties

Method

**Set and clear previous value**

The method to set an entity, like cleaning the old one, etc..

Strip tags **no**

Remove the tags or not.

Trim **no**

Trims the field value or not.

Field name

**title**

The machine name of the field, that should be changed. This property supports tokens. Example: body.value

Save entity **no**

Saves the entity or not after setting the value.

Field value

**[api\_person:name]**

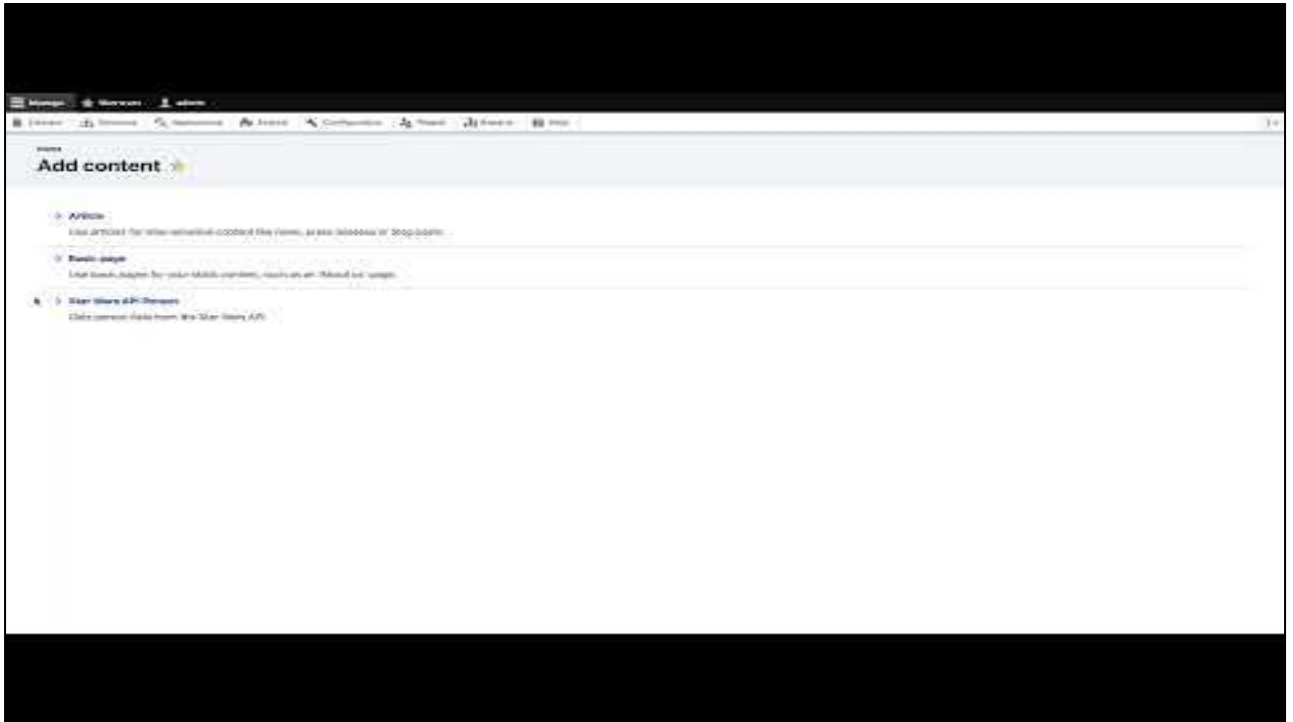
The new field value. This property supports tokens.

Entity

Provide the token name of the entity that this action should operate with.

Then I add an “Entity: set field value” action for each field that I want to update, using the “api\_person” token.





\* Click image to play video \*

Now let's see the model in action. I create a new Star Wars Person and I'm asked to select a name. I pick Darth Vader and click save and you can see that the other fields have been filled in, like Vaders height of 202 cm. If I edit and change the person to Leia, the information will be updated to match and the height is now 150 cm.

---

## ECA + HTTP Client Manager

As an administrator, I want to be notified by SMS when anyone logs into my account, so that I can react to potential security breaches

The second example I have will utilize a new API from twilio. Let's pretend you were tasked with building the following user story: As an administrator, I want to be notified by SMS when anyone logs into my account, so that I can react to potential security breaches.

Home > Administration > Configuration > Web services > HTTP Client Manager

Twilio ☆

- All Commands -

SendSMS

Send SMS

HTTP Method	URI	Operations
POST	Accounts/(AccountSid)/Messages.json	Configured Requests (0)

Parameters

Name	Type	Default	Description	Required	Location
AccountSid	String		Twilio Account SID	Yes	uri
MessagingServiceSid	String		Twilio Messaging Service SID	Yes	formParam
To	String		Phone number	Yes	formParam
Body	String		SMS Body	Yes	formParam

First I've created an HTTP client to interface with the Twilio API. Twilio can do a lot of things in the customer engagement field, but for this example I'm only using one operation called send SMS. There are four required parameters including twilio account information, the message to send and the phone number to send it to.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

## Notify admin of login ECA Model ☆

The diagram illustrates an Event-Condition-Action (ECA) model. It starts with a circle representing the event 'User login'. An arrow labeled 'is administrator?' points to a rounded rectangle labeled 'AND'. From the 'AND' box, an arrow labeled 'has phone number?' points to another rounded rectangle labeled 'Load account sid'. This is followed by a rounded rectangle labeled 'Load messaging sid'. A line from the bottom of the 'Load messaging sid' box branches down and then left to a rounded rectangle labeled 'Load phone number'. This is followed by a rounded rectangle labeled 'Load site name', then another labeled 'Load IP', and finally a rounded rectangle labeled 'Send SMS'. To the left of the diagram is a vertical toolbar with icons for hand, zoom, pan, and other editing tools.

This model has some similarities to the last one, the biggest difference being that I'm sending data to an API instead of getting data.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

The diagram shows a BPMN process starting with an event 'User login'. This event leads to an AND gateway. From the AND gateway, the process splits into two parallel paths. The top path consists of three tasks: 'Load account sid', 'Load messaging sid', and 'Load phone number'. The bottom path consists of three tasks: 'Load site name', 'Load IP', and 'Send SMS'. The two paths merge at the AND gateway.

LOGIN OF A USER  
User login

General

Template [Applied](#)

BPMN.IO

The event here is “login of a user” which fires for all users.

Home Management Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; UserLogin((User login)) --> IsAdmin{is administrator?}; IsAdmin --> AND{AND}; AND --> LoadAccount[Load account sid]; AND --> LoadMessaging[Load messaging sid]; AND --> LoadPhone[Load phone number]; AND --> LoadSite[Load site name]; AND --> LoadIP[Load IP]; LoadAccount --> SendSMS[Send SMS]; LoadMessaging --> SendSMS; LoadPhone --> SendSMS; LoadSite --> SendSMS; LoadIP --> SendSMS;
```

ROLE OF CURRENT USER is administrator?

General

Template Applied

Custom properties

User role Administrator

Negate the condition no

Negates the condition. Makes TRUE to FALSE and vice versa.

BPMN.IO

I only want to notify admins so I add a “role of current user” condition to check that.

Home Management Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; Start((User login)) --> AND{AND}; AND --> LoadPhone[Load phone number]; AND --> HasPhone{has phone number?}; LoadPhone --> LoadSite[Load site name]; LoadSite --> LoadIP[Load IP]; LoadIP --> SendSMS[Send SMS]; HasPhone --> LoadAccount[Load account sid]; LoadAccount --> LoadMessaging[Load messaging sid];
```

ENTITY: FIELD VALUE IS EMPTY  
has phone number?

General >

Template Applied >

Custom properties >

Field name  
field\_phone\_number  
The field name of the entity to check, if its value is empty.

Negate the condition  
yes  
Negates the condition. Makes TRUE to FALSE and vice versa.

Entity  
Provide the token name of the entity that this condition should operate with.

BPMN.IO

I need a phone number to send the SMS to so I add “Entity: field value is empty” condition to check if the admin user that just logged in has a phone number set.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; Start((User login)) --> AND1{AND}; AND1 --> LoadAccount[Load account sid]; LoadAccount --> AND2{AND}; AND2 --> LoadPhone[Load phone number]; AND2 --> LoadSite[Load site name]; LoadPhone --> LoadIP[Load IP]; LoadSite --> LoadIP; LoadIP --> SendSMS[Send SMS];
```

**CONFIG: READ**  
Load account sid

General

Template **Applied**

Custom properties

Config name  
**twilio\_http\_client.settings**  
The config name, for example system.site

Config key  
**account\_sid**  
The config key, for example page.front.  
Leave empty to use the whole config.

Include overridden  
**yes**  
Whether to apply module and settings.php overrides to values.

Name of token  
**account\_sid**  
The targeted configuration value will be loaded into this specified token.

BPMN.IO

The twilio API requires authentication so the next steps are to get that information from Drupal config storage with two “config: read” actions.



Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

```

    graph LR
      UserLogin((User login)) --> AND{AND}
      AND -- "has phone number?" --> LoadAccount[Load account sid]
      AND --> LoadMessaging[Load messaging sid]
      AND --> LoadPhone[Load phone number]
      LoadPhone --> LoadSite[Load site name]
      LoadSite --> LoadIP[Load IP]
      LoadIP --> SendSMS[Send SMS]
  
```

ENTITY: GET FIELD VALUE  
Load phone number

General

Template Applied

Custom properties

Field name  
**field\_phone\_number**  
The machine name of the field, that holds the value. This property supports tokens.

Name of token  
**phone\_number**  
The field value will be loaded into this specified token.

Entity  
Provide the token name of the entity that this action should operate with.

BPMN.IO

I can load the phone number from the user with a “entity: get field value” action. ~~Since I started with a user event, the action knows that it should read the “field\_phone\_number” field from the user entity.~~

**Correction:** You must tell ECA to load the data from the user entity. Do that by entering “user” in the “Entity” field.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home > Administration > Configuration > Workflow > Configure ECA - Events, Conditions, Actions

### Notify admin of login ECA Model ☆

Save Close [Browse available tokens.](#)

```
graph LR; Start((User login)) --> AND{AND}; AND --> LoadAccount[Load account sid]; AND --> LoadMessaging[Load messaging sid]; AND --> LoadPhone[Load phone number]; AND --> LoadSite[Load site name]; AND --> LoadIP[Load IP]; LoadAccount --> LoadMessaging; LoadPhone --> LoadSite; LoadSite --> LoadIP; LoadIP --> SendSMS[Send SMS];
```

**TWILIO - SEND SMS**  
Send SMS

General >

Template **Applied** >

Custom properties >

Twilio Account SID  
[account\_sid]

Twilio Messaging Service SID  
[messaging\_sid]

Phone number  
[phone\_number]

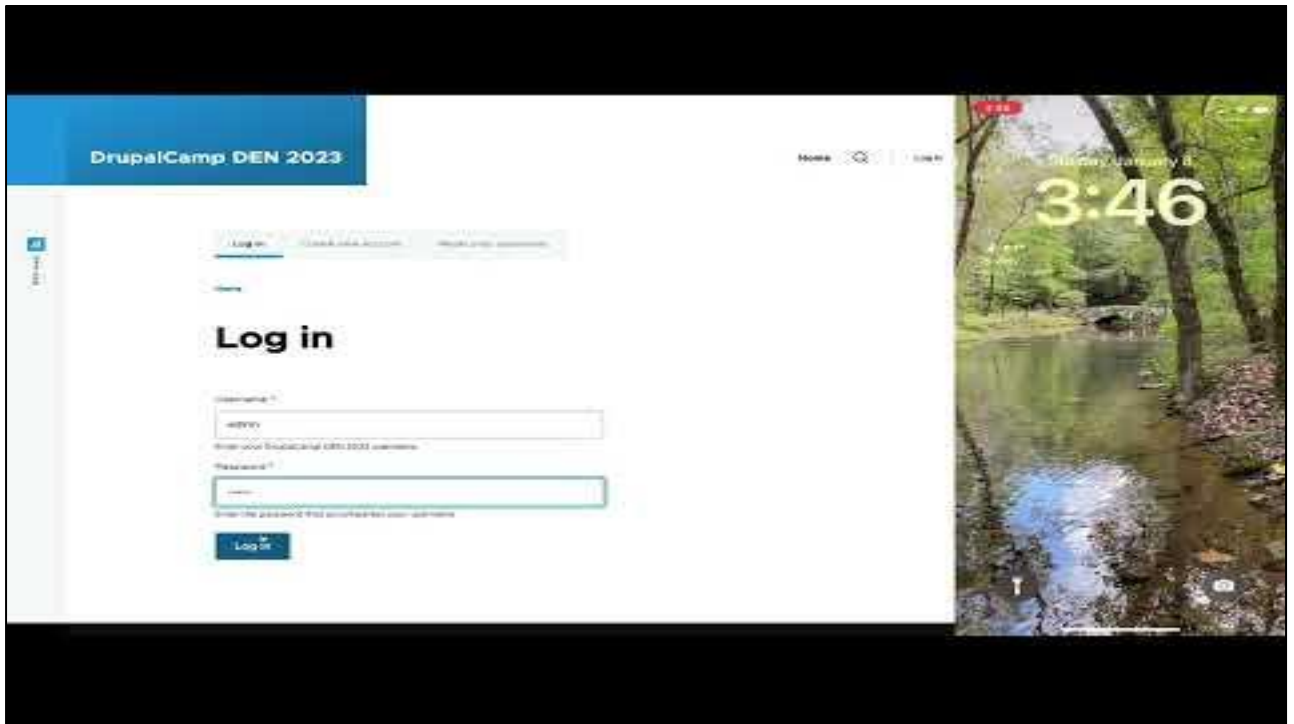
SMS Body  
[site\_name]: New login to your user "[u]

Replace tokens  
yes

When enabled, tokens will be replaced before executing the action. Please note: Actions might already take care of replacing tokens on their own. Therefore, use this option only with care and when it makes sense.

BPMN.IO

I added some more actions to gather more information, but after that's all done there is enough information to send an SMS message. All the parameters that were required by the API definition can be set by passing the tokens that were made from the previous actions so that twilio knows what to send and who to send it to.



\* Click image to play video \*

With our ECA model in place I can now login to the site with an admin account. In background ECA is processing the model and I should get a SMS shortly after logging in. And here it is: New login to your user "admin" from IP 10.99.99.1

---

## Recap

- ECA

Powerful processing engine to replace and improve upon the Rules module for Drupal

- HTTP Client Manager

Create and manage full featured API clients in a simple language like YAML or JSON

The combination of ECA and HTTP Client Manager is a powerful new solution for Drupal site builders and developers

The examples I've shown today are just the tip of the iceberg of what's possible. Hopefully I've shown you how powerful a processing engine ECA is, not just as a replacement for the rules module but as a new tool to solve problems that previously required custom modules; and how you can create and manage full featured api clients without writing any PHP code by defining them in a simpler language like YAML.

Developers, and especially site builders, can combine the power of these modules to build sites faster and better than before.



---

Thanks for watching!



---

**Brandon Williams**

Software Engineer  
rocketeerbkw.com

- Star Wars API example [https://dgo.re/http\\_client\\_swapi](https://dgo.re/http_client_swapi)
- ECA model examples  
[https://github.com/rocketeerbkw/eca\\_http-client-manager\\_examples](https://github.com/rocketeerbkw/eca_http-client-manager_examples)



Thanks for watching! If you'd like to get in touch with me I keep an updated list of my social media accounts on my website at [rocketeerbkw.com](http://rocketeerbkw.com). If you'd like to see the full star wars api client, I created a module for that on [drupal.org](http://drupal.org). If you'd like to import the ECA models I used in my examples to play with on your own site I've exported them and put them up on my github.

I'd like thank the organizers for having me and hope y'all enjoy the camp!