Electronic Certificate System

API Reference Manual

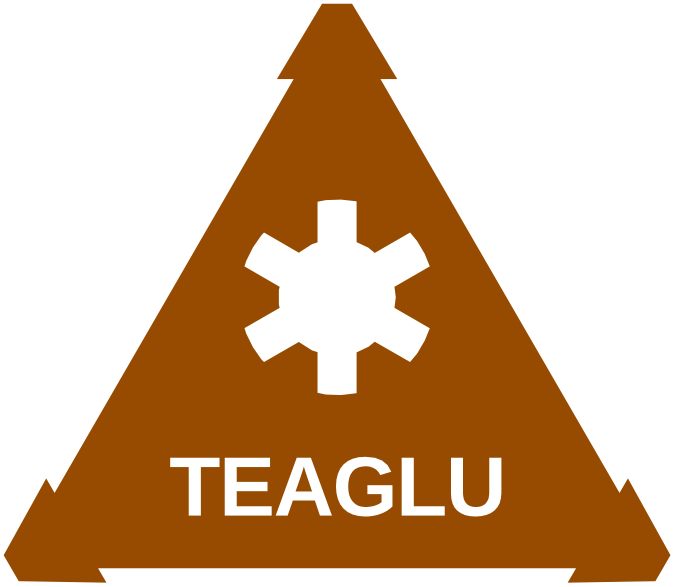

Table of Contents

Document Scope and Audience....4
Quickstart....4
    Integration....4
    Upload....4
    Example Command Line Call....5
API Usage....5
  Authentication....5
    Authentication by API Token....5
    Authentication by Session....6
  Endpoints....6
    Record Type Endpoints....6
    Certificate Options Endpoint....6
    Integration Create Endpoint....7
    Integration Upload Endpoint....8
  Call Patterns....9
    Retrieving a Record....9
    Querying for Records....9
    Creating a New Records....9
    Updating an Existing Record....9
    Deleting an Existing Record....9
  Response Code Classes....9
    2xx – Success....9
    4xx – Client Error....9
    5xx – Server Error....10
  Response Codes....10
    200 – Request Succeeded....10
    201 – Entity Created....10
    400 – Invalid Request....10
    403 – Access Denied....10
    404 – Object Not Found....10
    405 – Not Implemented....10
    409 – Referenced Record....10
    422 – Validation Error....11
    429 – Rate Limit Violation....11
    440 – No Session....11
    500 – Bugcheck or Internal Error....11
    503 – Unable to Access Resource....11
Data Dictionary....11
  Record Types....11
    Template....11
    Supplier....12
    Source....12

Certification....13
Product....13
Form Factor....14
Certificate....15
Relevant Implementation Details....16
Point-In-Time Fields....16
Down-Labeling Certificates....16

Document Scope and Audience

This document serves as the API reference for the Teaglu Electronic Certificate System (ECS). It is intended for developers who need to integrate with ECS from external systems and are expected to have a bachelor's degree in computer science or equivalent practical experience.

Quickstart

The most common integration is for a supplier’s line-of-business application to create or upload certificates. The difference between these two flows is:

Integration	**Upload**
Certificate number allocated by ECS	Certificate number allocated by supplier
API called before customer leaves	API called after order or delayed
Certificate created by ECS	Supplier created their own certificate

Integration

For the integration case, each call should:

1) Send a POST request to https://{domain}/api/integration/create with the relevant data

2) If the HTTP response code is 201, then the response will be the PDF record of the created certificate. The Location header will give a reference to the created record including it’s record ID.

3) If the HTTP response code is 422, then the response will be a JSON array of validation errors.

4) For any other HTTP response code, see the section on response codes.

Upload

1) Send a POST request to https://{domain}/api/integration/upload with the relevant data.

2) If the HTTP response code is 201, then the response will be a JSON representation of the created record. No further action is needed.

3) If the HTTP response code is 422, then the response will be a JSON array of validation errors.

4) For any other HTTP response code, see the section on response codes.

Example Command Line Call

This command executed at a linux command line will create a complete certificate and return the PDF in a single call:

```
curl -X POST "https://demo.growercert.com/api/integration/create" \
  -H "Authorization: Bearer 8c556715ba141a4ecbf79a63226971c7" \
  -H "Content-Type: application/json" \
  -H "Accept: application/octet-stream" \
  --data-binary @- \
  -o response.pdf <<'EOF'
{
	"sourceName": "AL2",
	"productName": "TifTuf® Bermuda",
	"formFactorName": "Sod",
	"certificationName": "Certified",
	"harvestDate": "2026-02-06",
	"printedDate": "2026-02-02",
	"amount": 42,
	"soldTo": "Test Delete Me",
	"certificateData": {
		"invoice": "1234",
		"shipper": "slow boat"
	}
}
EOF
```

API Usage

Authentication

Authentication by API Token

API authentication can be done in a stateless manner by including a bearer token with each request – this is the normal method of using the API outside of a browser.

A bearer token may be used by including it as an authorization header on each HTTP request, as shown below:

Authorization: Bearer 12358q7u98a7s9sd8f79879

API tokens can be created by the agency on the user management screen by creating a user and assigning a token.

Authentication by Session

A cookie-based session may be created by using the following endpoint:

https://{domain}/api/session

The following parameters should be passed to the endpoint in a POST request to create a new session:

Parameter	***Type***	***Definition***
loginName	string	The login name of a user record.
password	string	The password of the user record.
tz	string	The timezone the session should use. If this parameter is not passed, “America/New_York” will be used.

A session may be deleted by sending a DELETE request to the session endpoint.

Endpoints

The application is designed so that the web front-end uses the same endpoints as the API, so API functionality can be verified by using the application and viewing the API calls in the browser debugger interface.

Record Type Endpoints

The following endpoint correspond to data types as defined in the data dictionary. The {domain} portion of the URL is based on the DNS name assigned to the instance you are accessing.

Data Type	***Verbiage***	***Endpoint***
Template	Form	*https://{domain}/api/cert/template/*
Supplier	Grower	*https://{domain}/api/cert/supplier/*
Source	Field	*https://{domain}/api/cert/source/*
Certification	Certification	*https://{domain}/api/cert/certification/*
Certificate	Certificate	*https://{domain}/api/cert/certificate/*
Product	Variety	*https://{domain}/api/cert/product/*
Form Factor	Form Factor	*https://{domain}/api/cert/formfactor/*

Certificate Options Endpoint

The following GET endpoint will return all the valid options as an array of objects:

https://{domain}/api/cert/certificate/options

The following may be passed as GET parameters to limit the scope:

Name	***Type***	***Description***
supplierId	integer	Primary key of supplier
sourceId	integer	Primary key of source
productId	integer	Primary key of product
certificationId	integer	Primary key of certification

Each possibility record will be returned:

Name	***Type***	***Description***
supplierId	integer	Primary key of supplier
sourceId	integer	Primary key of source
productId	integer	Primary key of product
certificationId	integer	Primary key of certification
available	float	Amount of product available for sale in the listed combination, in the sales units (normally square feet)
cutoff	date	Date of the last approval / inspection contributing to the available amount

Integration Create Endpoint

The following POST endpoint allows a grower system to create a certificate in a single POST call, without having to worry about resolving IDs external to their system:

https://{domain}/api/integration/create

Name	***Type***	***Description***
supplierName	string	Primary key of supplier
sourceName	string	Full source name
productName	string	Full product name
certificationName	string	Full certification name
formFactorName	string	Full form factor name
templateName	string	Full template name

harvestDate	date	Date of harvest
amount	double	Amount of sale
soldTo	string	Name of person sold to
certificateData	JSON object	Custom attributes

If the call data fails validation rules, the endpoint will return a 422 response code, and the body of the response will be a JSON array of validation errors.

If the certificate is successfully created, the endpoint will return a 201 response code, and the body of the response will be the PDF of the newly created certificate.

Integration Upload Endpoint

The following POST endpoint allows a grower system to create a certificate in a single POST call, without having to worry about resolving IDs external to their system:

https://{domain}/api/integration/upload

Name	***Type***	***Description***
supplierName	string	Primary key of supplier
sourceName	string	Full source name
productName	string	Full product name
certificationName	string	Full certification name
formFactorName	string	Full form factor name
templateName	string	Full template name
harvestDate	date	Date of harvest
amount	double	Amount of sale
soldTo	string	Name of person sold to
certificateNo	string	Certificate number
certificateData	JSON object	Custom attributes

If the call data fails validation rules, the endpoint will return a 422 response code, and the body of the response will be a JSON array of validation errors.

If the certificate is successfully created, the endpoint will return a 201 response code, and the body of the response will be a JSON object corresponding to the created record.

Call Patterns

All API endpoints follow a normalized REST pattern, and send and receive data in JSON. Request bodies should include the JSON directly – it should not be encapsulated in form encoding.

Retrieving a Record

For each endpoint you can send a GET request to {endpoint}/id with the primary ID of the record. The response will be a JSON object with all object fields, or 404 if the record does not exist.

Querying for Records

For each endpoint you can send a GET request to {endpoint}/ and pass any field name as search parameters. The response will be an array of JSON objects with all object fields, or an empty array if there are no matches.

Creating a New Records

For each endpoint you can send a POST request to {endpoint}/ to create a new object. The request body should be a JSON object with appropriate fields. Do not include the id attribute. The response will be a JSON object with all fields including the allocated id attribute.

Updating an Existing Record

For each endpoint you can send a PUT request to {endpoint}/id to update an object. The request body should be a JSON object with fields to update. The response will be a JSON object with all fields included.

Deleting an Existing Record

For each endpoint you can send a DELETE request to {endpoint}/id to delete the object.

Response Code Classes

2xx – Success

2xx errors indicate that whatever you were trying to accomplish worked.

4xx – Client Error

4xx errors indicate an error that the application believes is on your end, so the same request should not be retried. If you believe your request is correct then please contact support and we can track down the root cause together.

5xx – Server Error

5xx errors indicate a severe code error or an infrastructure failure. Our AWS infrastructure is designed to self-heal so these may be transient, and we have alarms on our AWS Application Load Balancers to notify us of 5xx responses.

Response Codes

200 – Request Succeeded

This response code is returned if the request succeeded, and no more specific code applies.

201 – Entity Created

This response code is returned if the request resulted in the creation of a new entity.

400 – Invalid Request

This response code is returned if the endpoint is unable to correctly interpret the request. Typically this means the request is not a valid JSON object when one is expected.

403 – Access Denied

This response code is returned if the caller is attempting to access a resource they do not have permission to access.

404 – Object Not Found

This response code is returned if the caller is attempting to access an existing resource that does not exist. This may also be returned if the caller is attempting to access a valid resource, but the caller does not have access rights to the resource – this is done so that the response code does not leak the IDs of valid resources.

405 – Not Implemented

This response code is returned if the caller attempts to perform an action on a resource that the resource does not support.

409 – Referenced Record

This response code is returned if the caller attempts to delete a resource that cannot be deleted because it is referenced by another resource.

422 – Validation Error

This response code is returned if the requested action would cause the underlying resource to violate one of its integrity constraints.

429 – Rate Limit Violation

This response code is returned if the caller is attempting to call a rate-limited endpoint too often.

440 – No Session

This response code is returned if the caller is attempting a call using an invalid session, or if the call does not include a valid session token or bearer token.

500 – Bugcheck or Internal Error

This response code is returned if the endpoint has an unexpected error. If you receive a 500 response code from your call, please contact support.

503 – Unable to Access Resource

This response code is returned if the endpoint cannot access a required back-end resource such as a database or application server.

If you receive a 503 response code with our managed service, this may be a temporary error – but if the error persists please contact support.

Data Dictionary

Record Types

Template

The template record corresponds to a form layout.

Field Name	***Data Type***	***Description***
id	integer	Primary key
name	string	Name
baseFileId	integer	Primary key of the asset record holding the PDF the form is laid on top of.
definition	JSON	Field layout information
numberLength	integer	Length of the allocation

		certificate numbers
numberStart	integer	Starting number for allocation

Supplier

A supplier record corresponds to the entity selling the product and handing out the certificate. This would normally be a grower.

Field Name	***Data Type***	***Description***
id	integer	Primary key
numberPrefix	string	Prefix for certificate numbers. If you include the year in your numbers, you should change this value to start allocating numbers from the new sequence
numberStart	integer	Number to start at for new number allocations. This may be left blank to use 1, or used to skip number ranges.
address1	string	Address line 1
address2	string	Address line 2
city	string	City
state	string	State
zip	string	Zip Code
officePhone	string	Office phone number
signatureName	string	Signature to be placed on certificates, if there is no signature assigned to the user creating the certificate.
supplierData	JSON	Extension fields assigned to the supplier

Source

A source corresponds to the location where the product being certified was obtained. This is normally a field under certification.

Field Name	***Data Type***	***Description***

id	integer	Primary key
supplierId	integer	Foreign key of the supplier who manages this source
active	boolean	If the source is active
name	string	Name
sourceData	JSON	Extension fields assigned to the source

Certification

A certification corresponds to a certification level the product was inspected to. This is typically Certified, Registered, or Foundation.

Field Name	***Data Type***	***Description***
id	integer	Primary key
defaultTemplateId	integer	The default template to be used for certificates issued under this certification.
amountMultiplier	float	A multiplier to update the amount of product consumed. This is normally 1.
amountUnlimited	boolean	If true, do not count this certification level against the supply
certificationData	JSON	Extension fields assigned to the certificatin level
allowedTemplateId	integer[]	Alternate templates that can be assigned to the certification level. This is typically used to down-label product.

Product

A product record corresponds to the SKU or unique identifier of the item being sold under certification. This is common a variety.

Field Name	***Data Type***	***Description***
id	integer	Primary key

active	boolean	Active
name	string	Name
amountMultiplier	float	A multiplier for the amount of product consumed. This is normally 1.
amountUnlimited	float	If true, do not count this certification level against the supply
productData	JSON	Extension fields assigned to the product record.

Form Factor

Form factor records correspond to ways the same underlying product is sold. Common examples would be square feet, bins, bushels, or sprigs.

Field Name	***Data Type***	***Description***
id	integer	Primary key
name	string	Name
unitsName	string	Unit name the form factor is measured in
unitsAbbreviation	string	An abbreviation for the units the form factor is measured in
enableMultiplier	boolean	Enable a multiplier for this form factor
multiplierUnitsName	string	Unit name the multiplier is measured in
multiplierUnitsAbbreviation	string	An abbreviation for the unit name the multiplier this form factor is measured in
amountMultiplier	float	Numerator for the conversion from the form factor units to the approval units (normally acres)
amountDivisor	float	Denominator for the conversion from the form factor units to the approval units (normally acres)

Certificate

A certificate record corresponds to a certificate issued. Creation requests should supply the foreign keys, and point-in-time fields will be populated by the system.

Retrieving the PDF file can be done after record creation using the following endpoint:

https://{domain}/api/cert/certificate/{id}/pdf

Field Name	***Data Type***	***Description***
id	integer	Primary key
supplierId	integer	Foreign key of the supplier
sourceId	integer	Foreign key of the source
productId	integer	Foreign key of the product
certificationId	integer	Foreign key of the certification
templateId	integer	Foreign key of the template
formFactorId	integer	Foreign key of the form factor
certificateNo	string	Allocated certificate number
createdUserId	integer	Foreign key of the creating user
createdDate	date/time	Date and time of creation
printedDate	date/time	Date and time of printing. If this value is not populated the PDF will be labeled as draft.
voidDate	date	Date of void, or null if not voided. If this value is populated the PDF will be labeled as void.
voidReason	string	Reason for void, or null if not voided.
voidReplacementNo	string	Certificate number this voided certificate is replaced by. Not normally used.
supplierName	string	Supplier name at the time of creation.
sourceName	string	Source name at the time of creation.
productName	string	Product name at the time of creation.
certificationName	string	Certification name at the time of creation.

harvestDate	date	Date of harvest.
soldTo	string	Person or entity the sale was made to.
amount	float	Amount of the sale
amountMultiplier	float	Multiplier amount if used
supplierData	JSON	Supplier data at the time of creation
sourceData	JSON	Source data at the time of creation
productData	JSON	Product data at the time of creation
certificationData	JSON	Certification data at the time of creation
certificateData	JSON	Extension fields at the certificate level
formFactorData	JSON	Form factor data at the time of creation

Relevant Implementation Details

This sections lists implementation details that may affect your usage, or for general information.

Point-In-Time Fields

Because the certificate record represents a physical document, the values needed to recreate that document are stored as part of the certificate record. This ensures that any subsequent updates to the source records do not change the certificate.

For example, if a grower changes their name for marketing purposes, certificates issued prior to that change will still appear as they did when created.

Down-Labeling Certificates

The application requires the certification level selected to match the level of the approval, and will not issue a certified if an approval at the same certification level is not present.

Agencies can allow certificates to be printed on alternate forms, and some agencies have used this technique to allow their end users to pull from one certification level but print it as a lower certification level.

In this case, the certification level sent must match the certification level of the approval, but you can pass a template to have it print on the form for a different level.