

Documentation for the DigiCert Retail API

Oct 22, 2026

Table of Contents

Introduction	2
<i>API Basics</i>	2
<i>Authentication</i>	2
<i>Usage Guidelines</i>	2
<i>Requests</i>	2
<i>Responses</i>	3
<i>Endpoint Overview</i>	5
Sample Code	6
<i>Reissue Certificate</i>	6
<i>Retrieve Certificate</i>	9
Errors and Troubleshooting	11
Appendix	12
<i>Country Codes</i>	12
<i>Server Types</i>	15
<i>Code Signing Certificate Platforms</i>	16
<i>EV Code Signing Certificate Platforms</i>	17
<i>ISO 639-2 Language Codes</i>	18

Introduction

API Basics

All API requests are submitted via REST-based URLs using REST features including HEADER-based authentication and JSON/XML request types.

The requests will be accepted through port 443, which is the default port for SSL. The character set encoding of data in API calls and responses is UTF-8. To ensure a well-formed request, make sure that the USER-AGENT and CONTENT-LENGTH headers are specified in the request.

Authentication

The REST API requires access to the existing DigiCert APIs and utilizes the same keys. However to authenticate with the service, it uses **Basic HTTP Authentication over SSL**.

As basic authentication is an established standard, it is easy to implement. Using cURL:

```
curl -u "account_id:api_key" ...
```

Building the header manually:

1. Combine the account id and api key into a string: `account_id:api_key`
2. Base64 encode the string
3. Add the encoded string to the header: `Authorization: Basic {BASE64_ENCODED_STRING}`

So this is also valid in cURL:

```
curl -H "Authorization: Basic {BASE64_ENCODED_STRING}"
```

Usage Guidelines

Rate limits are put in place to prevent abuse, and may be changed in the future without notice. If you exceed the per-hour or per-day limit or otherwise abuse the service, the DigiCert Rest API may stop working for you temporarily.

DigiCert will typically rate limit API usage at around 1000 hits per hour. If you require a higher API limit threshold, contact a member of the DigiCert sales team to see what options are available for your usage patterns.

Requests

Requests consist of two or three parts: Method, Endpoint, and Body. Every request has a method and endpoint, but some also include a body. The body is formatted based on the CONTENT-TYPE header sent.

Method

The API uses four standard HTTP methods and several custom methods. The standard methods supported are GET, HEAD, OPTIONS, and POST. The custom methods are APPROVE, REISSUE, REJECT, RENEW, and REVOKE. The client can use any of these methods directly. This is the preferred method. To use with cURL:

```
curl -u "account_id:api_key" -X {CUSTOM_METHOD}...
```

So, to revoke a request:

```
curl -u "account_id:api_key" -X REVOKE ...
```

Some client libraries may not allow custom HTTP methods. For these libraries, there is an additional way to set the method for a request: use a custom HTTP header.

To use:

1. Set the HTTP method as POST.
2. Create a header named X-HTTP-METHOD-OVERRIDE and set the value to be one of the custom methods.

To use with cURL:

```
curl -u "account_id:api_key" -X POST -H "X-HTTP-Method-Override: REVOKE"...
```

Endpoint

REST requests use endpoints to retrieve and alter data. Endpoints are generally split into two types:

1. *Collection*: Represents a group of items. Examples include `/order` where POST creates a new item and GET returns a list of items.
2. *Item*: Represents a single item. These endpoints usually have an id attached at the end of the URL. Examples include `/order/00112233` where POST affects the particular order and GET returns details about the order.

Specific endpoints will be described later when explaining how to make specific requests.

Request Filters

Filters are appended to the end of a URL using a `?`. Multiple filters can be applied to the same request. If using multiple filters, they are separated with a `'&'` character.

Body and Content Type

Most requests require passing in either JSON or XML formatted data. Some requests may require standard content types like `image/jpeg`. If an endpoint supports or requires a format other than JSON or XML, it will be noted in the documentation. Generally GET requests are the only ones that do not require sending formatted data. With some endpoints, GET requests will support passing in additional parameters to filter the results.

For requests that require additional data, instead of using the generic `application/json` and `application/xml` content types, the API supports custom content types that also include a version number.

The following content types are currently supported:

1. `application/vnd.digicert.rest-v1+xml`
2. `application/vnd.digicert.rest-v1+json`

To use, create a header:

```
Content-Type: application/vnd.digicert.rest-v1+json
```

A full cURL request may be something like:

```
curl -u "account_id:api_key" -X POST \
-H "Content-Type: application/vnd.digicert.rest-v1+json" \
--data '{"foo":"bar"}' https://api.digicert.com/fez/00112233
```

Responses

Responses consist of two parts: Headers and Body. The body is formatted based on the content type requested in the ACCEPT header.

Headers

Every response will include an appropriate HTTP header. This header is based off of the [RFC 2616](#) specification.

Code	Description
200	General success response

Code	Description
201	Created: Useful for creation of requests, orders, etc
204	No Content: For successful requests that don't require a response
301	Moved Permanently: Returned in the unlikely event that a URL has changed. Will also return a LOCATION header with new URL. Clients should resubmit this request and submit future requests to this new URL
302	Moved Temporarily: Returned in the unlikely event that a URL has changed temporarily. Will also return a LOCATION header with new URL. Clients should resubmit this single request to this new URL
304	Content not modified: Useful when accessing a URL while waiting for a response. Only used if an IF-NONE-MATCH header was passed
400	General client error
401	Unauthorized: Returned if the page is accessed without a valid account/API Key combination
403	User doesn't have permission to perform the requested action
404	Returned if the page doesn't exist or the API doesn't have permission to interact with a particular item
405	Method not found. Returned if an endpoint doesn't support the method that was passed. It will also return an ALLOW header with supported methods
406	If the client requests an incompatible response (i.e. the client wants a v1 response for an endpoint only available in v2+), or the client doesn't specify a valid acceptable content-type
429	Too many requests. The client has sent too many requests in a given amount of time.
500	Unexpected behavior that the API couldn't recover from
503	The system is currently unavailable

For most of the GET requests, the API will also return an ETAG header so that clients can also pass in an IF-NONE-MATCH header.

Body and Content Type

For simplicity, the API support the same content types for both responses and requests.

For GET requests, the client does not send additional data to the API. For these requests, it is more appropriate to send an ACCEPT header than a CONTENT-TYPE header. However, if a CONTENT-TYPE header is passed and not an ACCEPT header (or the header is Accept: */*), the API will default to using the value set in the CONTENT-TYPE.

For other requests, a client only needs to send an ACCEPT header if the request is sent in one format and the response should be sent in a different one (i.e. the request is a JSON-encoded string, but the response needs to be in XML).

If the API cannot determine which format the client wants, it will return a 406 header.

Successful Response

All success responses will return an HTTP code in the 200 – 399 range. Clients receiving these headers will know that their request was successfully handled and will expect an appropriate response.

As an example, a client submits a POST request to the `/enterprise/certificate/ssl` endpoint. It will expect a `request_id` that could be used in subsequent Approval Requests. It would expect a response similar to below:

Documentation for the DigiCert Retail API		5
JSON (application/vnd.digicert.rest-v1+json)	XML (application/vnd.digicert.rest-v1+xml)	
<pre>{ "request_id":7902 }</pre>	<pre><response> <request_id>7902</request_id> </response></pre>	

Error Response

Error responses will return an HTTP code in 400+ range. Clients receiving these headers know that their request failed for various reasons. The API should always at least return a 500 error if there was a fatal error that wasn't caught. Most error responses return a special content type, based on the initial type being requested: `application/vnd.digicert.error+xml` for XML based requests, and `application/vnd.digicert.error+json` for JSON based. This content type is based on emerging standards.

For instance, a client wants to reject a pending request, but forgot to include the required NOTE. They would make a REJECT request to the `/request/7902` endpoint, and would receive a 400 response with the following body:

JSON (application/vnd.digicert.error+json)	XML (application/vnd.digicert.error+xml)
<pre>{ "errors":[{ "code":"required_param:note", "description":"Note is required" }] }</pre>	<pre><errors> <error> <code>required_param:note</code> <description>Note is required</description> </error> </errors></pre>

Endpoint Overview

HEAD Requests

Any endpoint that supports a GET method also supports the HEAD method. As defined by [RFC 2616](#), a HEAD request is treated the same as a GET request, except that it doesn't contain a response body.

OPTIONS Requests

All endpoints support an OPTIONS method call. The OPTIONS method as defined in [RFC 2616](#) returns the same ALLOW header as a 405 error response.

Sample Code

Reissue Certificate

For the examples below:

Account ID 001007
API Key n9hq0fp9q63htpmt7xcthztt5n4zx721
Order ID 111222

Using cURL

User creates a JSON string (named *data.json* for simplicity and formatted for readability):

```
{
  "server_type":2,
  "csr":"-----BEGIN CERTIFICATE REQUEST-----\n[CSR GOES HERE]\n-----END CERTIFICATE
REQUEST-----"
}
```

User uses curl from the commandline to send the REISSUE request:

```
curl -u "001007:n9hq0fp9q63htpmt7xcthztt5n4zx721" -X REISSUE \
-H "Content-Type: application/vnd.digicert.rest-v1+json" \
--data @data.json https://api.digicert.com/order/111222
```

Using C#

```
using System;
using System.Text;
using System.Net;
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string post_data = @"{
""server_type"":2,
""csr"":""-----BEGIN CERTIFICATE REQUEST-----\n[CSR GOES HERE]\n-----END
CERTIFICATE REQUEST-----"
            }";

            byte[] postBytes = Encoding.UTF8.GetBytes(post_data);
            string uri = "https://api.digicert.com/order/111222";

            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
            request.KeepAlive = false;
            request.ProtocolVersion = HttpVersion.Version10;
            request.Method = "REISSUE";
            request.UserAgent = "DigiCertAPIClient/0.1";
            request.ContentType = "application/vnd.digicert.rest-v1+json";
            request.ContentLength = postBytes.Length;
```

```
request.Headers.Add("Authorization", "Basic " + GetEncodedCredential());

try
{
    Stream requestStream = request.GetRequestStream();
    requestStream.Write(postBytes, 0, postBytes.Length);
    requestStream.Close();

    HttpResponseMessage response = (HttpResponseMessage)request.GetResponse();//throws
    WebException
    Console.WriteLine(new
    StreamReader(response.GetResponseStream()).ReadToEnd());
}
catch (WebException e)
{
    Console.WriteLine(e.Message);
}
}

static private String GetEncodedCredential()
{
    return Convert.ToBase64String(ASCIIEncoding.ASCII.GetBytes("001007" + ":" +
    "n9hq0fp9q63htpmt7xcthztt5n4zx721"));
}
}
```

Using PHP

```
$arr = array(
    "server_type" => 2,
    "csr" => "-----BEGIN CERTIFICATE REQUEST-----\n[CSR GOES HERE]\n-----END
    CERTIFICATE REQUEST-----");

$json = json_encode($arr);
$headers = array(
    'Content-Type: application/vnd.digicert.rest-v1+json'
);

$url = 'https://api.digicert.com/order/111222';

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_USERPWD, '001007:n9hq0fp9q63htpmt7xcthztt5n4zx721');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'REISSUE');
curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

$result = curl_exec($ch);
$http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
```

Using Ruby

```
require 'net/http'
require 'uri'

uri = URI.parse 'https://api.digicert.com/order/111222'

request = Net::HTTP::POST.new uri.request_uri
request.basic_auth('001007', 'n9hq0fp9q63htpmt7xcthztt5n4zx721')

request['Content-Type'] = 'application/vnd.digicert.rest-v1+json'
request['X-HTTP-Method-Override'] = 'REISSUE'

#Note that this JSON body is formatted for readability
request.body = <<JSON
{
  "server_type":2,
  "csr":"-----BEGIN CERTIFICATE REQUEST-----\n[CSR GOES HERE]\n-----END CERTIFICATE REQUEST-----"
}
JSON

http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true

result = http.request(request)
```

Retrieve Certificate

For the examples below:

Account ID 001007
API Key n9hq0fp9q63htpmt7xcthztt5n4zx721
Order ID 111222

GET requests are very simple. After authenticating with the service, having the client library access the URL is all that is required.

Using cURL

```
curl -u "001007:n9hq0fp9q63htpmt7xcthztt5n4zx721" \  
-H "Accept: application/vnd.digicert.rest-v1+json" \  
https://api.digicert.com/order/111222/certificate
```

Using C#

```
using System;  
using System.Text;  
using System.Net;  
using System.IO;  
  
namespace ConsoleApplication1  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string uri = "https://api.digicert.com/order/111222/certificate";  
  
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);  
            request.KeepAlive = false;  
            request.ProtocolVersion = HttpVersion.Version10;  
            request.Method = "GET";  
            request.UserAgent = "DigiCertAPIClient/0.1";  
            request.ContentType = "application/vnd.digicert.rest-v1+json";  
            request.Headers.Add("Authorization", "Basic " + GetEncodedCredential());  
  
            try  
            {  
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();//throws  
                Console.WriteLine(new  
                StreamReader(response.GetResponseStream()).ReadToEnd());  
            }  
            catch (WebException e)  
            {  
                Console.WriteLine(e.Message);  
            }  
        }  
    }  
  
    static private String GetEncodedCredential()
```

```
{
    return Convert.ToBase64String(ASCIIEncoding.ASCII.GetBytes("001007" + ":" +
    "n9hq0fp9q63htpmt7xcthztt5n4zx721"));
}
}
```

Using PHP

```
$url = 'https://api.digicert.com/order/111222/certificate';

$headers = array(
    'Accept: application/vnd.digicert.rest-v1+json',
);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_USERPWD, '001007:n9hq0fp9q63htpmt7xcthztt5n4zx721');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

$result = curl_exec($ch);
$http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
```

Using Ruby

```
require 'net/http'
require 'uri'

uri = URI.parse 'https://api.digicert.com/order/111222/certificate'

request = Net::HTTP::Get.new uri.request_uri
request.basic_auth('001007', 'n9hq0fp9q63htpmt7xcthztt5n4zx721')

request['Accept'] = 'application/vnd.digicert.rest-v1+json'

http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true

result = http.request request
```

Errors and Troubleshooting

As mentioned in the introduction, error responses will return an HTTP code in 400+ range. Clients receiving these headers know that their request failed for various reasons. The API should always at least return a 500 error if there was a fatal error that wasn't caught. Most error responses return a special content type, based on the initial type being requested:

`application/vnd.digicert.error+xml` for XML based requests, and

`application/vnd.digicert.error+json` for JSON based. This content type is based on emerging standards.

In the case of 406 errors, the API doesn't know what format the client requested, so it will send `application/vnd.digicert.error+json`.

Error Return Values

Parameter Name	Data Type	Description
errors	[collection]	A collection of error codes
error	[set]	A data set containing information about a specific error
code	[string]	An error code that identifies the error that occurred
description	[string]	A human readable description of the error

Sample XML Error Response

For instance, a client wants to reject a pending request, but forgot to include the required NOTE. They would make a REJECT request to the `/request/7902` endpoint, and would receive a 400 response with the following body:

JSON (<code>application/vnd.digicert.error+json</code>)	XML (<code>application/vnd.digicert.error+xml</code>)
<pre>{ "errors": [{ "code": "required_param:note", "description": "Note is required" }] }</pre>	<pre><errors> <error> <code>required_param:note</code> <description>Note is required</description> </error> </errors></pre>

Appendix

Country Codes

USA	us
Canada	ca
United Kingdom	gb
Afghanistan	af
Åland Islands	ax
Albania	al
Algeria	dz
American Samoa	as
Andorra	ad
Angola	ao
Anguilla	ai
Antarctica	aq
Antigua and Barbuda	ag
Argentina	ar
Armenia	am
Aruba	aw
Australia	au
Austria	at
Azerbaijan	az
Bahamas	bs
Bahrain	bh
Bangladesh	bd
Barbados	bb
Belarus	by
Belgium	be
Belize	bz
Benin	bj
Bermuda	bm
Bhutan	bt
Bolivia	bo
Bonaire, Sint Eustatius and Saba	bq
Bosnia and Herzegovina	ba
Botswana	bw
Bouvet Island	bv
Brazil	br

British Indian Ocean Territory	io
Brunei Darussalam	bn
Bulgaria	bg
Burkina Faso	bf
Burundi	bi
Cambodia	kh
Cameroon	cm
Cape Verde	cv
Cayman Islands	ky
Central African Republic	cf
Chad	td
Chile	cl
China	cn
Christmas Island	cx
Cocos (Keeling) Islands	cc
Colombia	co
Comoros	km
Congo	cg
Congo, the Democratic Republic of the	cd
Cook Islands	ck
Costa Rica	cr
Côte d'Ivoire	ci
Croatia	hr
Curaçao	cw
Cyprus	cy
Czech Republic	cz
Denmark	dk
Djibouti	dj
Dominica	dm
Dominican Republic	do
Ecuador	ec
Egypt	eg
El Salvador	sv
Equatorial Guinea	gq
Eritrea	er

Estonia	ee
Eswatini	sz
Ethiopia	et
Falkland Islands (Malvinas)	fk
Faroe Islands	fo
Fiji	fj
Finland	fi
France	fr
French Guiana	gf
French Polynesia	pf
French Southern Territories	tf
Gabon	ga
Gambia	gm
Georgia	ge
Germany	de
Ghana	gh
Gibraltar	gi
Greece	gr
Greenland	gl
Grenada	gd
Guadeloupe	gp
Guam	gu
Guatemala	gt
Guernsey	gg
Guinea	gn
Guinea-Bissau	gw
Guyana	gy
Haiti	ht
Heard and McDonald Islands	hm
Holy See	va
Honduras	hn
Hong Kong	hk
Hungary	hu
Iceland	is
India	in

Indonesia	id	Moldova	md	S. Georgia and S. Sandwich Isls.	gs
Iraq	iq	Monaco	mc	Saint Barthélemy	bl
Ireland	ie	Mongolia	mn	Saint Helena, Ascension and Tristan da Cunha	sh
Isle of Man	im	Montenegro	me	Saint Kitts & Nevis	kn
Israel	il	Montserrat	ms	Saint Lucia	lc
Italy	it	Morocco	ma	Saint Martin	mf
Jamaica	jm	Mozambique	mz	Saint Pierre and Miquelon	pm
Japan	jp	Myanmar	mm	Saint Vincent and The Grenadines	vc
Jersey	je	Namibia	na	Samoa	ws
Jordan	jo	Nauru	nr	San Marino	sm
Kazakhstan	kz	Nepal	np	Sao Tome and Principe	st
Kenya	ke	Netherlands	nl	Saudi Arabia	sa
Kiribati	ki	New Caledonia	nc	Senegal	sn
Korea, South	kr	New Zealand	nz	Serbia	rs
Kuwait	kw	Nicaragua	ni	Seychelles	sc
Kyrgyzstan	kg	Niger	ne	Sierra Leone	sl
Laos	la	Nigeria	ng	Singapore	sg
Latvia	lv	Niue	nu	Sint Maarten	sx
Lebanon	lb	Norfolk Island	nf	Slovakia	sk
Lesotho	ls	Northern Mariana Islands	mp	Slovenia	si
Liberia	lr	Norway	no	Solomon Islands	sb
Libya	ly	Not Assigned	xx	Somalia	so
Liechtenstein	li	Oman	om	South Africa	za
Lithuania	lt	Pakistan	pk	South Sudan	ss
Luxembourg	lu	Palau	pw	Spain	es
Macao	mo	Palestinian Territory	ps	Sri Lanka	lk
Macedonia, Republic of North	mk	Panama	pa	Sudan	sd
Madagascar	mg	Papua New Guinea	pg	Suriname	sr
Malawi	mw	Paraguay	py	Svalbard and Jan Mayen Islands	sj
Malaysia	my	Peru	pe	Sweden	se
Maldives	mv	Philippines	ph	Switzerland	ch
Mali	ml	Pitcairn Island	pn	Taiwan	tw
Malta	mt	Poland	pl	Tajikistan	tj
Marshall Islands	mh	Portugal	pt	Tanzania	tz
Martinique	mq	Puerto Rico	pr	Thailand	th
Mauritania	mr	Qatar	qa	Timor-Leste	tl
Mauritius	mu	Réunion	re	Togo	tg
Mayotte	yt	Romania	ro		
Mexico	mx	Russia	ru		
Micronesia, Federated States of	fm	Rwanda	rw		

Tokelau	tk	U.S. Minor Outlying Islands	um	Vietnam	vn
Tonga	to	Uganda	ug	Virgin Islands, British	vg
Trinidad and Tobago	tt	Ukraine	ua	Virgin Islands, U.S.	vi
Tunisia	tn	United Arab Emirates	ae	Wallis and Futuna Islands	wf
Turkey	tr	Uruguay	uy	Western Sahara	eh
Turkmenistan	tm	Uzbekistan	uz	Yemen	ye
Turks and Caicos Islands	tc	Vanuatu	vu	Zambia	zm
Tuvalu	tv	Venezuela	ve	Zimbabwe	zw



Server Types

Apache	2	Microsoft IIS 10	70
BEA Weblogic 8 & 9	42	Microsoft IIS 5 or 6	14
Barracuda	41	Microsoft IIS 7	40
Bea Weblogic 7 and older	29	Microsoft IIS 8	67
Cisco	30	Microsoft Live Communications Server 2005	37
Citrix (Other)	39	Microsoft Lync Server 2010	59
Citrix Access Essentials	46	Microsoft Lync Server 2013	69
Citrix Access Gateway 4.x	50	Microsoft OCS R2	60
Citrix Access Gateway 5.x and higher	58	Microsoft Office Communications Server 2007	38
F5 Big-IP	31	Microsoft Small Business Server 2008 & 2011	62
F5 FirePass	32	Netscape Enterprise Server	15
IBM HTTP Server	7	Netscape iPlanet	9
Java Web Server (Javasoftware / Sun)	10	Novell NetWare	17
Juniper	33	Novell iChain	65
Lighttpd	44	OTHER	-1
Lotus Domino	11	Oracle	18
Mac OS X Server	49	Qmail	34
Microsoft Exchange Server 2003	47	SunOne	35
Microsoft Exchange Server 2007	36	Tomcat	24
Microsoft Exchange Server 2010	48	WebStar	26
Microsoft Exchange Server 2013	68	Zeus Web Server	28
Microsoft Exchange Server 2016	71	cPanel	43
Microsoft Forefront Unified Access Gateway	66	nginx	45
Microsoft IIS 1.x to 4.x	13		

Code Signing Certificate Platforms

Adobe AIR	52
Apple OS X	53
Microsoft Authenticode	51
Microsoft Office VBA	54
Mozilla	56
Other	57
Sun Java	55



EV Code Signing Certificate Platforms

EV Code Signing Certificates must be installed on to a hardware token device. DigiCert offers two provisioning options:

1. We can ship you a hardware token with your EV Code Signing Certificate already provisioned for you (`cs_provisioning_method = 'ship_token'`)
2. If you have a qualified hardware token (see chart below), you can provision your own certificate using the tool found at <https://www.digicert.com/StaticFiles/DigiCertHardwareCertificateInstaller.zip> (`cs_provisioning_method = 'client_app'`)

AEP Keyper	15
ARX PrivateServer	16
Bull Trustway Crypto PCI	17
FIPS 140-2 Level 2 or Common Criteria EAL4+	-1
SafeNet IDPrime 930	22
SafeNet eToken 5100	6
SafeNet eToken 5105	7
SafeNet eToken 5110	19
SafeNet eToken 5110 CC	23
SafeNet eToken 5110 FIPS	20
SafeNet eToken 5110+ CC (940B) (ECC P-256 only)	25
SafeNet eToken 5110+ CC (940C)	26
SafeNet eToken 5110+ FIPS	24
SafeNet eToken 5200	8
SafeNet eToken 5205	9
SafeNet eToken PRO 72K	3
SafeNet eToken PRO Anywhere	2
SafeNet iKey 4000	10
Safenet Luna	12
Thales nShield	13
Utimaco CryptoServer	14
ePass3003	21

ISO 639-2 Language Codes

Abkhazian	abk	Bosnian	bos	Dzongkha	dzo
Achinese	ace	Braj	bra	Eastern Frisian	frs
Acoli	ach	Breton	bre	Efik	efi
Adangme	ada	Buginese	bug	Ekajuk	eka
Adyghe	ady	Bulgarian	bul	English	eng
Afar	aar	Buriat	bua	Erzya	myv
Afrikaans	afr	Burmese	mya	Estonian	est
Ainu	ain	Caddo	cad	Ewe	ewe
Akan	aka	Catalan	cat	Ewondo	ewo
Albanian	sqi	Cebuano	ceb	Fang	fan
Aleut	ale	Central Khmer	khm	Fanti	fat
Amharic	amh	Chamorro	cha	Faroese	fao
Angika	anp	Chechen	che	Fijian	fij
Arabic	ara	Cherokee	chr	Filipino	fil
Aragonese	arg	Cheyenne	chy	Finnish	fin
Arapaho	arp	Chinese	zho	Fon	fon
Arawak	arw	Chinook	chn	French	fra
Armenian	hye	Chipewyan	chp	Friulian	fur
Assamese	asm	Choctaw	cho	Fulah	ful
Asturian	ast	Chuukese	chk	Ga	gaa
Avaric	ava	Chuvash	chv	Galibi Carib	car
Awadhi	awa	Cornish	cor	Galician	glg
Aymara	aym	Corsican	cos	Ganda	lug
Azerbaijani	aze	Cree	cre	Gayo	gay
Balinese	ban	Creek	mus	Gbaya	gba
Baluchi	bal	Crimean Tatar	crh	Georgian	kat
Bambara	bam	Croatian	hrv	German	deu
Basa	bas	Czech	ces	Gilbertese	gil
Bashkir	bak	Dakota	dak	Gondi	gon
Basque	eus	Danish	dan	Gorontalo	gor
Beja	bej	Dargwa	dar	Grebo	grb
Belarusian	bel	Delaware	del	Guarani	grn
Bemba	bem	Dhivehi	div	Gujarati	guj
Bengali	ben	Dinka	din	Gwich'in	gwi
Bhojpuri	bho	Dogri	doi	Haida	hai
Bikol	bik	Dogrib	dgr	Haitian	hat
Bilin	byn	Duala	dua	Hausa	hau
Bini	bin	Dutch	nld	Hawaiian	haw
Bislama	bis	Dyula	dyu	Hebrew	heb

Herero	her	Komi	kom	Manchu	mnc
Hiligaynon	hil	Kongo	kon	Mandar	mdr
Hindi	hin	Konkani	kok	Mandingo	man
Hiri Motu	hmo	Korean	kor	Manipuri	mni
Hmong	hmn	Kosraean	kos	Manx	glv
Hungarian	hun	Kpelle	kpe	Maori	mri
Hupa	hup	Kuanyama	kua	Mapudungun	arn
Iban	iba	Kumyk	kum	Marathi	mar
Icelandic	isl	Kurdish	kur	Mari	chm
Igbo	ibo	Kurukh	kru	Marshallese	mah
Iloko	ilo	Kutenai	kut	Marwari	mwr
Inari Sami	smn	Ladino	lad	Masai	mas
Indonesian	ind	Lahnda	lah	Mende	men
Ingush	inh	Lamba	lam	Mi'kmaq	mic
Inuktitut	iku	Lao	lao	Minangkabau	min
Inupiaq	ipk	Latvian	lav	Mirandese	mwł
Irish	gle	Lezghian	lez	Greek	ell
Italian	ita	Limburgan	lim	Mohawk	moh
Japanese	jpn	Lingala	lin	Moksha	mdf
Javanese	jav	Lithuanian	lit	Mongo	lol
Judeo-Arabic	jrb	Low German	nds	Mongolian	mon
Judeo-Persian	jpr	Lower Sorbian	dsb	Mossi	mos
Kabardian	kbd	Lozi	loz	N'Ko	nqo
Kabyle	kab	Luba-Katanga	lub	Nauru	nau
Kachin	kac	Luba-Lulua	lua	Navajo	nav
Kalaallisut	kal	Luiseno	lui	Ndonga	ndo
Kalmyk	xal	Lule Sami	smj	Neapolitan	nap
Kamba (Kenya)	kam	Lunda	lun	Nepali	nep
Kannada	kan	Luo	luo	Newari	new
Kanuri	kau	Lushai	lus	Nias	nia
Kara-Kalpak	kaa	Luxembourgish	ltz	Niuean	niu
Karachay-Balkar	krc	Macedo-Romanian	rup	Nogai	nog
Karelian	krl	Macedonian	mkd	North Ndebele	nde
Kashmiri	kas	Madurese	mad	Northern Frisian	frr
Kashubian	csb	Magahi	mag	Northern Sami	sme
Kazakh	kaz	Maithili	mai	Norwegian	nor
Khasi	kha	Makasar	mak	Norwegian Bokmål	nob
Kikuyu	kik	Malagasy	mlg	Norwegian Nynorsk	nno
Kimbundu	kmb	Malay	msa	Nyamwezi	nym
Kinyarwanda	kin	Malayalam	mal	Nyanja	nya
Kirghiz	kir	Maltese	mlt	Nyankole	nyn

Nyoro	nyo	Sichuan Yi	iii	Tlingit	tli
Nzima	nzi	Sicilian	scn	Tok Pisin	tpi
Occitan	oci	Sidamo	sid	Tokelau	tkl
Ojibwa	oji	Siksika	bla	Tonga (Nyasa)	tog
Oriya	ori	Sindhi	snd	Tonga (Tonga Islands)	ton
Oromo	orm	Sinhala	sin	Tsimshian	tsi
Osage	osa	Skolt Sami	sms	Tsonga	tso
Ossetian	oss	Slave (Athapaskan)	den	Tswana	tsn
Palauan	pau	Slovak	slk	Tumbuka	tum
Pampanga	pam	Slovenian	slv	Turkish	tur
Pangasinan	pag	Somali	som	Turkmen	tuk
Panjabi	pan	Soninke	snk	Tuvalu	tvl
Papiamento	pap	South Ndebele	nbl	Tuvinian	tyv
Pedi	nso	Southern Altai	alt	Twi	twi
Persian	fas	Southern Sami	sma	Udmurt	udm
Pohnpeian	pon	Southern Sotho	sot	Uighur	uig
Polish	pol	Spanish	spa	Ukrainian	ukr
Portuguese	por	Sranan Tongo	srn	Umbundu	umb
Pushto	pus	Sukuma	suk	Upper Sorbian	hsb
Quechua	que	Sundanese	sun	Urdu	urd
Rajasthani	raj	Susu	sus	Uzbek	uzb
Rapanui	rap	Swahili	swa	Vai	vai
Rarotongan	rar	Swati	ssw	Venda	ven
Romanian	ron	Swedish	swe	Vietnamese	vie
Romansh	roh	Swiss German	gsw	Votic	vot
Romany	rom	Syriac	syr	Walloon	wln
Rundi	run	Tagalog	tgl	Waray	war
Russian	rus	Tahitian	tah	Washo	was
Samoan	smo	Tajik	tgk	Welsh	cym
Sandawe	sad	Tamashek	tmh	Western Frisian	fry
Sango	sag	Tamil	tam	Wolaytta	wal
Santali	sat	Tatar	tat	Wolof	wol
Sardinian	srd	Telugu	tel	Xhosa	xho
Sasak	sas	Tereno	ter	Yakut	sah
Scots	sco	Tetum	tet	Yao	yao
Scottish Gaelic	gla	Thai	tha	Yapese	yap
Selkup	sel	Tibetan	bod	Yiddish	yid
Serbian	srp	Tigre	tig	Yoruba	yor
Serer	srr	Tigrinya	tir	Zapotec	zap
Shan	shn	Timne	tem	Zaza	zza
Shona	sna	Tiv	tiv	Zenaga	zen

Documentation for the DigiCert Retail API				21	
Zhuang	zha	Zulu	zul	Default	zxx
		Zuni	zun		

