API Documentation

All API requests should have the following header: { Accept: application/json }

API Key Authentication:

All requests that require API Key authentication must have the following header in order to be authenticated: { Authorization: Bearer $api_key } where $api_key is the api key for the particular app

User Authentication:

Once a user is authenticated a JWT (JSON web token) will be returned, once a token is received all further requests must have the following header in order to be authenticated: { Authorization: Bearer $auth_token } where $auth_token is the JWT

***Authentication Note**: requests with more than 1 Authentication Method listed will accept any of the methods. Requests with 1 method listed will only accept that method.*

Staging Base URL: http://cms.iversoft.ca

Staging API Key:
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcHBfa2V5IjoiYmFzZTY0OlZvb2g1ZzdSNXBOQ29JT2VcL0J1N3JESVpSOEZiZVdpOHJaeU1LckFlV3d3PSIsImlzcyI6Imh0dHA6XC9cL2xvY2FsaG9zdCIsImlhdCI6MTQ4NzY4ODE4OSwiZXhwIjoxNDg3NzE2OTg5LCJuYmYiOjE0ODc2ODgxODksImp0aSI6IjBlYjgxMTkzNTk1Njk1YTFjZjkwNzUwMjMzNDc3ZTNlIn0.v0OW9CJt8Rq-vBMfIeCPVHSTUUlVsvIqZKN06MSmpK4"

URL	Description	Authentication Methods	Method	Parameters	Response

api/authenticate	Authentication request, must do this once before any other requests in order to receive a valid token. Token must be saved to use in all subsequent requests.	API Key	POST	--Regular User
				
				email: "jordan@iversoft.ca"
				
				password: "123456"
				
				--Google Auth
				
				google_auth_code: "a29.GlsVBH6oq..."
				
				--Facebook Auth
				
				facebook_access_token: "MgDUOsdqoBAL..."
					`token: "eyJ0eXAiOiJKV1QiL...",`
					`expires: "2017-02-01 03:13 UTC",`
					`user: {`
					`id: 1`
					`role: 1`
					`name: "Jordan Admin"`
					`email: "jordan@iversoft.ca"`
					`created_at: "2016-12-02 13:09:01"`
					`updated_at: "2016-12-02 13:40:26"`
					`}`

api/users/resetpassword	request to send the password reset email	API Key	POST	email: "jordan@iversoft.ca"	success: "Password reset sent successfully"

api/users/list	Request to get a list of users	API Key
		
		User
			GET	offset: return offset number
				
				limit: return limit amount
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					data: {
					0: {
					id: 1,
					role: 1,
					name: "Jordan Admin",
					email: "jordan@iversoft.ca",
					created_at: "2016-12-02 13:09:01"
					}
					}
					}

api/users/single/{id}	Request to get information for a user	API Key User	GET	User ID as parameter {id}	id: 1 role: 1 name: "Jordan Admin" email: "jordan@iversoft.ca" created_at: "2016-12-02 13:09:01"
api/users/new	Request to create a new user, returns a user object on success	API Key	POST	name: user name (string|max:255) email: user email (string|max:255|unique) password: user password (string|min:6)	id: 1 role: 1 name: "Jordan Admin" email: "jordan@iversoft.ca" created_at: "2016-12-02 13:09:01"
api/users/edit/{id}	Request to edit a user, returns the user object on success	User	POST	User ID as parameter {id} name: user name (string|max:255) email: user email (string|max:255|unique) password: new password (optional|min:6)	id: 1 role: 1 name: "Jordan Admin Edit" email: "jordan@iversoft.ca" created_at: "2016-12-02 13:09:01"

api/blog/list	Request to get a list of blog posts, if a user_id is specified it will return all posts authored by that user - both published and draft.
	
	if user_id is not specified all published posts will be returned.
		API Key
		
		User
			GET	offset: return offset number (int)
				
				limit: return limit amount (int)
				
				user_id: user id to grab posts by this author (int)
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					data: {
					0: {
					id: 501,
					author: 51,
					title: "New Blog Post 01",
					content: "<p><samp><small>Cupcake i…liquorice.</l
					created_at: "2017-01-09 17:53:43",
					updated_at: "2017-01-25 21:49:05",
					published: 1,
					views: 4
					image: {
					id: 1,
					blog_post_id: 3,
					file_location: "TBPMHrSSeG",
					created_at: "2016-12-23 16:46:39",
					updated_at: "2016-12-23 16:46:39",
					file_sizes: {
					original: {
					url: "http://localhost:8888/images/render/original
					HrSSeG",
					size: {
					width: 73,
					height: 80
					}
					},
					}
					}
					}
					}
					}
					}

api/blog/single/{id}	Request to get information for a blog post	API Key
		
		User
			GET	Blog Post ID as parameter {id}	id: 501
					author: 51
					title: "New Blog Post 01"
					content: "<p><samp><small>Cupcake i…liquorice.</l
					created_at: "2017-01-09 17:53:43"
					updated_at: "2017-01-25 21:49:05"
					published: 1
					views: 4
					image: {
					id: 1,
					blog_post_id: 3,
					file_location: "TBPMHrSSeG",
					created_at: "2016-12-23 16:46:39",
					updated_at: "2016-12-23 16:46:39",
					file_sizes: {
					original: {
					url: "http://localhost:8888/images/render/original
					HrSSeG",
					size: {
					width: 73,
					height: 80
					}
					},
					}
					}

api/blog/new	Request to create a new blog post, returns a blog post object on success	User	POST	title: Blog Post title (string|max:255)
				
				content: Blog Post content (string)
				
				published: published (boolean)
				
				picture: image file (optional)
					id: 501
					
					author: 51
					
					title: "New Blog Post 01"
					
					content: "<p><samp><small>Cupcake i…liquorice.</l
					
					created_at: "2017-01-09 17:53:43"
					
					updated_at: "2017-01-25 21:49:05"
					
					published: 1
					
					views: 0
					
					image: {
					id: 1,
					blog_post_id: 501,
					file_location: "TBPMHrSSeG",
					created_at: "2016-12-23 16:46:39",
					updated_at: "2016-12-23 16:46:39",
					file_sizes: {
					original: {
					url: "http://localhost:8888/images/render/original
					HrSSeG",
					size: {
					width: 73,
					height: 80
					}
					},
					}
					
					}

api/blog/edit/{id}	Request to edit a blog post, returns the blog post object on success	User	POST	Blog Post ID as parameter {id}
				
				title: Blog Post title (string|max:255)
				
				content: Blog Post content (string)
				
				published: published (boolean)(optional)
				
				picture: image file (optional)
					id: 501
					
					author: 51
					
					title: "New Blog Post 01"
					
					content: "<p><samp><small>Cupcake i…liquorice.</l
					
					created_at: "2017-01-09 17:53:43"
					
					updated_at: "2017-01-25 21:49:05"
					
					published: 1
					
					views: 0
					
					image: {
					id: 1,
					blog_post_id: 501,
					file_location: "TBPMHrSSeG",
					created_at: "2016-12-23 16:46:39",
					updated_at: "2016-12-23 16:46:39",
					file_sizes: {
					original: {
					url: "http://localhost:8888/images/render/original
					HrSSeG",
					size: {
					width: 73,
					height: 80
					}
					},
					}
					
					}

api/calendar/list	Request to get a list of calendar events	API Key
		
		User
			GET	offset: return offset number (int)
				
				limit: return limit amount (int)
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					data: {
					0: {
					
					id: 13,
					
					title: "test",
					
					all_day: null,
					
					start: "2017-01-10 08:00:00",
					
					end: "2017-01-10 10:00:00",
					
					content: "fgfdgfdgdfgdfg",
					
					recurring_parent: null,
					
					author: 1,
					
					created_at: "2017-01-10 12:47:28",
					
					updated_at: "2017-01-10 12:47:28"
					
					}
					}
					}

api/calendar/single/{id}	Request to get information for a single event	API Key User	GET	Event ID as parameter {id}	id: 13, title: "test", all_day: null, start: "2017-01-10 08:00:00", end: "2017-01-10 10:00:00", content: "fgfdgfdgdfgdfg", recurring_parent: null, author: 1, created_at: "2017-01-10 12:47:28", updated_at: "2017-01-10 12:47:28"

api/calendar/userview/{id?}	Request to get a list of calendar events for a specific user, includes display details to use with a calendar display format.
	
	If the user id value is NOT specified it will return events for the current authorized user
		User	GET	User_id as url param {id} (optional)	[
					
					0: {
					
					id: 8,
					
					title: "mine",
					
					all_day: null,
					
					start: "2016-12-05 08:00:00",
					
					end: "2016-12-05 09:00:00",
					
					content: "xdcvdxsv",
					
					recurring_parent: null,
					
					author: 1,
					
					created_at: "2016-12-14 14:27:33",
					
					updated_at: "2016-12-28 19:24:37",
					
					allDay: false,
					
					url: "http://localhost:8888/calendar/event/8",
					
					color: "#3a87ad",
					
					editable: true,
					
					}
					
					]

api/calendar/event/new	Request to create a new calendar event	User	POST	title: required|max:255 content: required allday: boolean datestart: required|date dateend: required|date	id: 13, title: "test", all_day: null, start: "2017-01-10 08:00:00", end: "2017-01-10 10:00:00", content: "fgfdgfdgdfgdfg", recurring_parent: null, author: 1, created_at: "2017-01-10 12:47:28", updated_at: "2017-01-10 12:47:28"
api/calendar/event/save/{id}	Request to update an existing calendar event	User	POST	Event_id as url param {id} title: max:255 content: string allday: boolean datestart: date dateend: date	id: 13, title: "test", all_day: null, start: "2017-01-10 08:00:00", end: "2017-01-10 10:00:00", content: "fgfdgfdgdfgdfg", recurring_parent: null, author: 1, created_at: "2017-01-10 12:47:28", updated_at: "2017-01-10 12:47:28"
api/calendar/event/delete/{id}	Request to delete an existing calendar event	User	DELETE	Event_id as url param {id}	"success"
api/device/register	Request to register a mobile device to a user If device exists already (device_token + production) then the device is updated, if it doesn't exist already then a new device is created. This request will always set the logged_out flag to false	User	POST	device_token: required|max:255 production: required|boolean platform: required|max:255 endpoint: required|max:255 device_name: max:255 os_version: max:255 app_version: max:255	id: 1, user_id: 1, device_token: "sdfsdffsdfsdfdsfsddsfsdfsdfsdffsd", production: flase, endpoint: "sdaasdasd", platform: "ios", device_name: null, os_version: null, app_version: null, logged_out: false, last_notification: null, created_at: "2017-02-02 14:57:49", updated_at: "2017-02-02 14:58:38"

api/device/logout	Request to logout of a user device This request will set the logged_out flag to true for the specified device	User	POST	device_token: required|max:255 production: required|boolean	id: 1, user_id: 1, device_token: "sdfsdffsdfsdfdsfsddsfsdfsdfsdffsd", production: false, endpoint: "sdaasdasd", platform: "ios", device_name: null, os_version: null, app_version: null, logged_out: true, last_notification: null, created_at: "2017-02-02 14:57:49", updated_at: "2017-02-02 14:58:38"

api/messaging/inbox	Request to retrieve message threads for current user	User	GET	offset: return offset number (int)
				
				limit: return limit amount (int)
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					data: {
					0: {
					
					"id": 2,
					"user_id": 1,
					"title": "Help!",
					"deleted_at": null,
					"created_at": "2017-03-29 13:57:39",
					"updated_at": "2017-04-05 19:40:59",
					"unread": false,
					"participants": [
					{
					"id": 1,
					"role": 1,
					"name": "Jordan Admin",
					"email": "jordan@iversoft.ca",
					"created_at": "2017-03-08 14:10:36",
					"updated_at": "2017-04-05 17:04:31",
					"gender": "male",
					"biography": "hjhgj",
					"date_of_birth": "1990-08-21",
					"pivot": {
					"message_threads_id": 2,
					"user_id": 1
					}
					},
					
					]
					}
					}
					}

api/messaging/trashed	Request to retrieve trashed message threads for current user	User	GET	offset: return offset number (int)
				
				limit: return limit amount (int)
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					data: {
					0: {
					
					"id": 2,
					"user_id": 1,
					"title": "Help!",
					"deleted_at": null,
					"created_at": "2017-03-29 13:57:39",
					"updated_at": "2017-04-05 19:40:59",
					"unread": false,
					"participants": [
					{
					"id": 1,
					"role": 1,
					"name": "Jordan Admin",
					"email": "jordan@iversoft.ca",
					"created_at": "2017-03-08 14:10:36",
					"updated_at": "2017-04-05 17:04:31",
					"gender": "male",
					"biography": "hjhgj",
					"date_of_birth": "1990-08-21",
					"pivot": {
					"message_threads_id": 2,
					"user_id": 1
					}
					},
					
					]
					}
					}
					}

api/messaging/thread/messages/{id}	Request to retrieve messages from a specific thread	User	GET	message thread id as url param {id}
				
				offset: return offset number (int)
				
				limit: return limit amount (int)
					{
					offset: 0,
					limit: 10,
					total: 10,
					remaining: 190,
					
					"data": [
					
					{
					
					"id": 61,
					
					"user_id": 1,
					
					"message_threads_id": 2,
					
					"body": "test2",
					
					"read": 0,
					
					"created_at": "2017-04-05 19:33:25",
					
					"updated_at": "2017-04-05 19:33:25",
					
					"photo": null
					
					},
					
					]
					}

api/messaging/unread	Request to retrieve number of current threads with unread messages	User	GET		{
					unread_threads: 2,
					}

api/messaging/message/new	Request to send new message, will create a new message thread if one does not exist between both users	User	POST	to_user: id of user to receive message (int required)
				
				subject: title of new thread if creating new thread (string)
				
				content: content string (string)
				
				photo: photo to attach (image)
					{
					"id": 2,
					"user_id": 1,
					"title": "Help!",
					"deleted_at": null,
					"created_at": "2017-03-29 13:57:39",
					"updated_at": "2017-05-11 17:15:44",
					"messages": [
					{
					"id": 3,
					"user_id": 1,
					"message_threads_id": 2,
					"body": "testestsetsetset",
					"read": 1,
					"created_at": "2017-03-29 13:56:33",
					"updated_at": "2017-04-05 12:36:59",
					"photo": null
					},
					}

api/messaging/message/reply/{id}	Request to send a reply message from within a message thread, no need to attach subject or to_user.
	
	If message thread is known, always use this over new message
		User	POST	message_thread id as url param {id}
				
				reply_content: content string (string required)
				
				photo: photo to attach (image)
					{
					"id": 2,
					"user_id": 1,
					"title": "Help!",
					"deleted_at": null,
					"created_at": "2017-03-29 13:57:39",
					"updated_at": "2017-05-11 17:15:44",
					"messages": [
					{
					"id": 3,
					"user_id": 1,
					"message_threads_id": 2,
					"body": "testestsetsetset",
					"read": 1,
					"created_at": "2017-03-29 13:56:33",
					"updated_at": "2017-04-05 12:36:59",
					"photo": null
					},
					}

api/messaging/restore/{id}	Restore a trashed message thread	User	POST	message_thread id as url param {id}	'result' : "message thread restored"

api/messaging/read/message/{id}	Mark an individual message as read	User	POST	message id as url param {id}	{ "id": 3, "user_id": 1, "message_threads_id": 2, "body": "testestsetsetset", "read": 1, "created_at": "2017-03-29 13:56:33", "updated_at": "2017-04-05 12:36:59", "photo": null }
api/messaging/read/thread/{id}	Mark an entire message thread as read from the current user's perspective	User	POST	message_thread id as url param {id}	{ "id": 2, "user_id": 1, "title": "Help!", "deleted_at": null, "created_at": "2017-03-29 13:57:39", "updated_at": "2017-05-11 17:15:44", "messages": [{ "id": 3, "user_id": 1, "message_threads_id": 2, "body": "testestsetsetset", "read": 1, "created_at": "2017-03-29 13:56:33", "updated_at": "2017-04-05 12:36:59", "photo": null }, }
api/messaging/delete/{id}	Trash a message thread	User	DELETE	message_thread id as url param {id}	'result' => "message thread deleted"