

顔認証コネクトデバイス

公開 WEB API 仕様書

バージョン : R15-10-v3.20

発行日 : 2023年10月

目次

1. 公開 WEB API 一覧表	1
2. 個人情報管理方法	1
3. WEB API 仕様	2
3.1. ログイン	2
3.2. 端末情報取得	3
3.3. アクセス履歴取得	4
3.4. アクセス履歴最終番号取得	6
3.5. アクセス履歴(サムネイル)取得	7
3.6. 勤怠情報入力履歴取得	8
3.7. 勤怠情報入力履歴最終番号取得	10
3.8. ユーザ登録	11
3.9. ユーザ登録情報取得	13
3.10. ユーザ登録情報修正	15
3.11. ユーザ削除	17
3.12. 複数ユーザ登録情報取得（最大 1000 件）	18
3.13. アクセス履歴最終番号更新メッセージ要求	20
3.14. 酒気帯び確認情報取得	21
3.15. 注文受付情報取得	23
4. API 要求仕様（クライアント PC 側での実装必要）	25
4.1. アクセス履歴最終番号更新メッセージ送信	25
5. その他	26
5.1. HTTP Header	26
5.2. Cookie 要否一覧	26
5.3. WEB API 使用例	26
5.4. 代表的なシーケンス図	27
5.4.1. ユーザ情報の取得①（リアルタイム/1 秒周期）	27
5.4.2. ユーザ情報の取得②（リアルタイム）	27
5.4.3. 勤怠情報の取得（1 回/日）	27
5.4.4. ユーザ情報の更新	28
5.4.5. ユーザ情報の取得（QR コード初期値）	28
5.5. サムネイル" encrypted_file"の復号方法	29
5.6. 顔写真" encrypted_file"の暗号化方法	29

1. 公開 WEB API 一覧表

#	項目	Method	URL	申請	詳細
1	ログイン	POST	/login	-	3.1.
2	端末情報取得	GET	/api/terminalinfo	-	3.2.
3	アクセス履歴取得	POST	/api/accesslog	-	3.3.
4	アクセス履歴最終番号取得	GET	/api/accesslog/id	-	3.4.
5	アクセス履歴(サムネイル)取得	GET	/api/{user#}/accesslog/thumbnail/{id}	要	3.5.
6	勤怠情報入力履歴取得	POST	/api/attendancelog	-	3.6.
7	勤怠情報入力履歴最終番号取得	GET	/api/attendancelog/id	-	3.7.
8	ユーザ登録	POST	/api/{user#}/user	要	3.8.
9	ユーザ登録情報取得	GET	/api/{user#}/user/{id}	要	3.9.
10	ユーザ登録情報修正	PUT	/api/{user#}/user/{id}	要	3.10.
11	ユーザ削除	DELETE	/api/{user#}/user/{id}	要	3.11.
12	複数ユーザ登録情報取得	GET	/api/{user#}/user/list/{page#}	要	3.12.
13	アクセス履歴最終番号更新メッセージ要求	POST	/api/message/accesslog	-	3.13.
14	酒気帯び確認情報取得	POST	/api/alcoholchecklog		3.14.
15	注文受付情報取得	POST	/api/orderreceptionlog		3.15.

メッセージ受信には、以下 API をクライアント PC 側で実装する必要があります。

#	項目	Method	URL	申請	詳細
1	アクセス履歴最終番号更新メッセージ送信	POST	{host_ip}/api/message/accesslog	-	4.1.

2. 個人情報管理方法

情報取り扱いに関する契約を締結後、割り振られた user# と AES128 用の Key と初期化ベクトル (IV) を配布します。

#	契約者	申請日	user# (8 char)	Key (string, 16 char)	IV (string, 16 char)

3. WEB API 仕様

3.1. ログイン

URL	Method	Request	Response
/login	POST	Content-Type: application/json; charset=UTF-8	
		{ "username": "admin", "password": "aaaa" }	<div>【OK】</div> <div>{ "result": true, "message": { "modelType": "face_tst", "temperatureCelsius": "C" } }</div> <div>【NG】</div> <div>{ "result": false, "message": "login.incorrect_username_or_password" }</div>

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	ユーザ名	username	string	"username": "admin"	
2	パスワード	password	string	"password": "aaaa"	

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	端末タイプ	modelType	string	"modelType": "face_tst"	
3	温度単位	temperatureCelsius	string	"temperatureCelsius": "C"	C: 摂氏、F: 華氏
4	エラーメッセージ	message	string	"login.connect_with_master_ip"	sub 端末にログインを試みている場合
				"login.incorrect_username_or_password"	username, password のいずれかに不具合ありの場合

3.2. 端末情報取得

URL	Method	Request	Response
/api/terminalinfo	GET	Content-Type: application/json; charset=UTF-8	
		N/A	<pre> 【OK】 { "result": true, "message": [{ "connect_yn": 1, "terminal_sn": "NZ53MZCBMO", "terminal_ip": "172.16.80.201", "terminal_name": "Sub1", "terminal_role": "Slave" }, { "connect_yn": 1, ... }] }</pre>

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	接続状態	connect_yn	integer	"connect_yn": 1	1:正常 0:異常
3	端末#	terminal_sn	string	"terminal_sn": "NZ53MZCBMO"	
4	端末 IP アドレス	terminal_ip	string	"terminal_ip": "172.16.80.201"	
5	端末名	terminal_name	string	"terminal_name": "Sub1"	
6	端末設定	terminal_role	string	"terminal_role": "Slave"	Master: メイン端末 Slave: サブ端末

3.3. アクセス履歴取得

URL	Method	Request	Response
/api/accesslog	POST	Content-Type: application/json; charset=UTF-8	
		<pre>{ "type": "4", "date": "2021-08-06 12:00:00", "id": "0" }</pre>	<p>【OK】</p> <pre>{ "result": true, "totalCount": 27, "message": [{ "access_type": 1, "created_at": "2021-08-06 12:02:33", "id": 152, "mask": 1, "name": "日立太郎", "pass": 1, "person_id": "246911", "qr": "53434f50414a...53539", "temperature": 36.4, "terminal_id": "70:4A:0E:17:E9:E0", "terminal_sn": "SCOPA8A27E" }, ...] }</pre> <p>【NG】</p> <pre>{ "result": false, "message": "An error has occurred." }</pre>

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	取得形式	type	string	"type": "4"	0: 全データ 1: 最終データ 2: date (UTC time) valid 3: id valid 4: date (local time) valid
2	アクセス日時	date	string	"date": "2021-08-06 00:00:00"	type=2 or 4 の時、有効、date 以降のアクセス履歴を取得 形式: YYYY-MM-DD HH:MM:SS
3	アクセス ID	id	string	"id": "0"	type=3 の時、有効、id 以降のアクセス履歴を取得

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	該当アクセス数	totalCount	integer	"totalCount": 27	
3	アクセス形式	access_type	integer	"access_type": 1	0: OFF 1: 顔認証 2: パスコード 3: QR
4	アクセス日時	created_at	string	"created_at": "2021-08-06 12:02:33"	
5	アクセス ID	id	integer	"id": 152	
6	マスク着用	mask	integer	"mask": 1	0: 非着用 / 1: 着用
7	ユーザ名	name	string	"name": "日立太郎"	
8	通過判定結果	pass	integer	"pass": 1	0: 否 / 1: 可
9	ユーザ ID	person_id	string	"person_id": "246911"	
10	QR コード	qr	string	"qr": "53434f50414a...53539"	HEX 表示, 文字列化する場合、UTF8 を使用
11	温度	temperature	double	"temperature": 36.4	
12	端末 ID	terminal_id	string	"terminal_id": "70:4A:0E:17:E9:E0"	無線 LAN MAC アドレス
13	端末 #	terminal_sn	string	"terminal_sn": "SCOPA8A27E"	
14	エラーメッセージ	message	string	"An error has occurred."	type, date, id のいずれかに 不具合ありの場合。もしくは例外 発生時。

3.4. アクセス履歴最終番号取得

URL	Method	Request	Response
/api/accesslog/id	GET	Content-Type: application/json; charset=UTF-8	
		N/A	【OK】 { "result": true, "id": 152 }

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	(最終)アクセス ID	id	integer	"id": 152	

3.5. アクセス履歴(サムネイル)取得

URL	Method	Request	Response
/api/{user#}/accesslog/thumbnail/{id}	GET	Content-Type: application/json; charset=UTF-8	
		N/A	<div>【OK】</div> <pre>{ "result": true, "message": { "encrypted_file": "WWuIuA...Ib0tk=" } }</pre> <div>【NG】</div> <pre>{ "result": false, "message": "accesslog.no_thumbnail" }</pre>

データ一覧 (URL)

#	項目	ラベル	データ型	例	説明
1	契約者#	user#			契約者別に提供
2	アクセス ID	id			1 から 100000 の整数

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	該当するサムネイルが無い場合、false をリターン
2	サムネイル	encrypted_file	string	"encrypted_file": "WWuIuA...Ib0tk="	Key, IV を用いて AES128 を復号 (5.3 参照)
3	エラーメッセージ	message	string	"accesslog.no_thumbnail"	画像が存在しない場合

3.6. 勤怠情報入力履歴取得

URL	Method	Request	Response
/api/attendancelog	POST	Content-Type: application/json; charset=UTF-8	
		<pre>{ "type": "1", "date": "0000-00-00 00:00:00", "id": "0" }</pre>	<p>【OK】</p> <pre>{ "result": true, "totalCount": 1, "message": [{ "access_type": 2, "created_at": "2021-08-06 12:02:33", "id": 152, "mask": 1, "name": "日立太郎", "pass": 1, "person_id": "246911", "status": 1, "temperature": 36.4, "terminal_id": "70:4A:0E:17:E9:E0", "terminal_no": 123, "terminal_sn": "SCOPA8A27E" }] }</pre> <p>【NG】</p> <pre>{ "result": false, "message": "An error has occurred." }</pre>

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	取得形式	type	string	"type": "1"	0: 全データ 1: 最終データ 2: date (UTC time) valid 3: id valid 4: date (local time) valid
2	アクセス日時	date	string	"date": "0000-00-00 00:00:00"	type=2 or 4 の時、有効、date 以降のアクセス履歴を取得 形式: YYYY-MM-DD HH:MM:SS
3	アクセス ID	id	string	"id": "0"	type=3 の時、有効、id 以降のアクセス履歴を取得

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	該当アクセス数	totalCount	integer	"totalCount": 1	
3	アクセス形式	access_type	integer	"access_type": 2	0: OFF 1: 顔認証 2: パスコード 3: QR
4	アクセス日時	created_at	string	"created_at": "2021-08-06 12:02:33"	
5	アクセス ID	id	integer	"id": 152	
6	マスク着用	mask	integer	"mask": 1	0: 非着用 / 1: 着用
7	ユーザ名	name	string	"name": "日立太郎"	
8	通過判定結果	pass	integer	"pass": 1	0: 否 / 1: 可
9	ユーザ ID	person_id	string	"person_id": "246911"	
10	選択勤怠情報	status	integer	"status": 1	1: 出勤 2: 退勤 3: 外出 4: 戻り
11	温度	temperature	double	"temperature": 36.4	
12	端末 IP アドレス	terminal_id	string	"terminal_id": "70:4A:0E:17:E9:E0"	無線 LAN MAC アドレス
13	端末 NO	terminal_no	integer	"terminal_no": 123	3 桁までの任意の番号
14	端末 #	terminal_sn	string	"terminal_sn": "SCOPA8A27E"	端末のシリアルナンバー
15	エラーメッセージ	message	string	"An error has occurred."	type, date, id のいずれかに 不具合ありの場合。もしくは例外 発生時。

3.7. 勤怠情報入力履歴最終番号取得

URL	Method	Request	Response
/api/attendancelog/id	GET	Content-Type: application/json; charset=UTF-8	
		N/A	【OK】 { "result": true, "id": 152 }

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	(最終)アクセス ID	id	integer	"id": 152	

3.8. ユーザ登録

URL	Method	Request	Response
/api/{user#}/user	POST	Content-Type: multipart/form-data	
		{ "encrypted_file": "WWuIuA...Ib0tk=", "person_id": "40000", "name": "日立太郎", "password": "0000", "division1": "日立 LG", "division2": "開発本部", "division3": "開発 2 部", "division4": "SW 開発チーム", "terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:" }	【OK】 { "result": true, "message": "" }
			【NG】 { "result": false, "message": "validate.encrypted_file is invalid." }

データ一覧 (URL)

#	項目	ラベル	データ型	例	説明
1	契約者#	user#			契約者別に提供

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	顔写真	encrypted_file	string	"encrypted_file": "WWuIuA...Ib0tk="	Key, IV を用いて AES128 暗号化 (5.4 参照)
2	ユーザ ID	person_id	string	"person_id": "40000"	英数字最大 10 文字 ="0"は登録不可
3	ユーザ名	name	string	"name": "日立太郎"	最大 22 文字
4	パスワード	password	string	"password": "0000"	4 桁の数字
5	組織 1	division1	string	"division1": "日立 LG"	最大 16 文字
6	組織 2	division2	string	"division2": "開発本部"	最大 16 文字
7	組織 3	division3	string	"division3": "開発 2 部"	最大 16 文字
8	組織 4	division4	string	"division4": "SW 開発チーム"	最大 16 文字
9	ユーザ登録端末	terminal_sn_list	string	"terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:"	

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	エラーメッセージ	message	string	"validate.encrypted_file is invalid."	画像の暗号化に不具合ありの場合
				"validate.a_person_id_that_already_exists"	指定した person_id が既に登録されている場合
				"validate.please_check_the_picture"	画像ファイルに不具合あり、特徴量を抽出できなかった場合
				"validate.a_person_that_already_exists"	画像ファイルから抽出した特徴量の人物が既に登録されている場合

3.9. ユーザ登録情報取得

URL	Method	Request	Response
/api/{user#}/user/{id}	GET	Content-Type: application/json; charset=UTF-8	
		N/A	<pre> 【OK】 { "result": true, "message": { "id": 1, "person_id": "40000", "name": "日立太郎", "division1": "日立 LG ", "division2": "開発本部", "division3": "開発 2 部", "division4": "SW 開発チーム", "qr": "", "mail_to": "", "notice": "", "terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:" } } </pre>

データ一覧 (URL)

#	項目	ラベル	データ型	例	説明
1	契約者#	user#			契約者別に提供
2	管理 ID	id			1 から 30000 の整数

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	管理 ID	id	integer	"id": 1	
3	ユーザ ID	person_id	string	"person_id": "40000"	
4	ユーザ名	name	string	"name": "日立太郎"	
5	組織 1	division1	string	"division1": "日立 LG"	
6	組織 2	division2	string	"division2": "開発本部"	
7	組織 3	division3	string	"division3": "開発 2 部"	
8	組織 4	division4	string	"division4": "SW 開発チーム"	
9	QR コード	qr	string	"qr": "53434f50414a...53539"	HEX 表示, 文字列化する場合、UTF8 を使用
10	メール送信先	mail_to	string	"mail_to": ""	
11	メッセージ	notice	string	"notice": ""	
12	ユーザ登録済端末 未	terminal_sn_list	string	"terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:"	

3.10. ユーザ登録情報修正

URL	Method	Request	Response
/api/{user#}/user/{id}	PUT	Content-Type: multipart/form-data	
id=1～		{ "encrypted_file": "WWuIuA...Ib0tk=", "person_id": "40000", "name": "日立太郎", "password": "1234", "division1": "日立 LG", "division2": "開発本部", "division3": "開発 2 部", "division4": "SW 開発チーム", "qr": "", "mail_to": "", "notice": "", "terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:" }	【OK】 { "result": true, "message": "" }
			【NG】 { "result": false, "message": "validate.encrypted_file is invalid." }

データ一覧 (URL)

#	項目	ラベル	データ型	例	説明
1	契約者 #	user#			契約者別に提供
2	管理 ID	id			1 から 30000 の整数

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	顔写真	encrypted_file	string	"encrypted_file": "WWuIuA...Ib0tk="	Key, IV を用いて AES128 暗号化 (5.4 参照)
2	ユーザ ID	person_id	string	"person_id": "40000"	変更不可
3	ユーザ名	name	string	"name": "日立太郎"	最大 22 文字
4	パスワード	password	string	"password": "1234"	4 桁の数字
5	組織 1	division1	string	"division1": "日立 LG"	最大 16 文字
6	組織 2	division2	string	"division2": "開発本部"	最大 16 文字
7	組織 3	division3	string	"division3": "開発 2 部"	最大 16 文字
8	組織 4	division4	string	"division4": "SW 開発チーム"	最大 16 文字
9	QR コード	qr	string	"qr": "53434f50414a...53539"	最大 35 文字、文字列を数値 (HEX) 表記
10	メール送信先	mail_to	string	"mail_to": ""	
11	メッセージ	notice	string	"notice": ""	
12	ユーザ登録端末	terminal_sn_list	string	"terminal_sn_list": "NZ53MJCY7T:NZ53MZCBMO:"	

データー一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	エラーメッセージ	message	string	"validate.encrypted_file is invalid."	画像の暗号化に不具合ありの場合
				"validate.a_person_id_that_already_exists"	指定した person_id が既に登録されている場合
				"validate.please_check_the_picture"	画像ファイルに不具合あり、特徴量を抽出できなかった場合
				"validate.a_person_that_already_exists"	画像ファイルから抽出した特徴量の人物が既に登録されている場合

3.11. ユーザ削除

URL	Method	Request	Response
/api/{user#}/user/{id}	DELETE	Content-Type: application/json; charset=UTF-8	
		update_method=web	【OK】 { "result": true, "message": "" }

データ一覧 (URL)

#	項目	ラベル	データ型	例	説明
1	契約者#	user#			契約者別に提供
2	管理 ID	id			1 から 30000 の整数、または"all"（一括削除）

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	

3.12. 複数ユーザ登録情報取得（最大 1000 件）

URL	Method	Request	Response
/api/{user#}/user/list/{page#}	GET	Content-Type: application/json; charset=UTF-8	
		N/A	<pre> 【OK】 { "result": true, "totalCount": 2, "message": [{ "division1": "日立 LG ", "division2": "営業本部", "division3": "営業 1 部", "division4": "ODD 販売チーム", "terminal_sn_list": "NZ53MZCBMO:", "id": 2, "name": "芝浦花子", "person_id": "40001", }, { "division1": "日立 LG ", "division2": "開発本部", : : }] }</pre>

データ一覧（URL）

#	項目	ラベル	データ型	例	説明
1	契約者#	user#			契約者別に提供
2	ページ#	Page#			0 から 29 の整数

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	該当ユーザ数	totalCount	integer	"totalCount": 2	
3	組織 1	division1	string	"division1": "日立 LG"	
4	組織 2	division2	string	"division2": "営業本部"	
5	組織 3	division3	string	"division3": "営業 1 部"	
6	組織 4	division4	string	"division4": "ODD 販売チーム"	
7	ユーザ登録済 端末	terminal_sn_list	string	"terminal_sn_list": "NZ53MZCBMO:"	
8	管理 ID	id	integer	"id": 2	
9	ユーザ名	name	string	"name": "芝浦花子"	
10	ユーザ ID	person_id	string	"person_id": "40001"	

3.13. アクセス履歴最終番号更新メッセージ要求

URL	Method	Request	Response
/api/message/accesslog	POST	Content-Type: application/json; charset=UTF-8	
		{ "type": "0", "host_ip": "172.16.80.1", "port": "7000" }	【OK】 { "result": true, "message": { "id": 152 } }
			【NG】 { "result": false, "message": "It is a lack of information." }

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	要求内容	type	string	"type": "0"	0: 送信 1: 解除
2	メッセージ送信先 IP アドレス	host_ip	string	"host_ip": "172.16.80.1"	type=0 の時、有効、メッセージ送信先は 最大 1、上書き設定
3	メッセージ送信先 ポート番号	port	string	"port": "7000"	設定は任意。初期値"80"。 type=0 の時、有効、メッセージ送信先は 最大 1、上書き設定。

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	(最終)アクセス ID	id	integer	"id": 152	
3	エラーメッセージ	message	string	"It is a lack of information."	type もしくは host_ip が指 定されていない場合
				"An error has occurred."	例外発生時

3.14. 酒気帯び確認情報取得

URL	Method	Request	Response
/api/alcoholchecklog	POST	Content-Type: application/json; charset=UTF-8	
		<pre>{ "type": "0", "date": "0000-00-00 00:00:00", "id": "0" }</pre>	<p>【OK】</p> <pre>{ "result": true, "totalCount": 27, "message": [{ "alco_detect": 0, "alco_val": 0.0, "car_num": "", "confirmed_at": "2023-06-21 8:46:49", "confirmor": "山田 拓美", "created_at": "2023-06-21 8:39:10", "driver": "日立 太郎", "fever_detect": 0, "id": 3, "instruction": "なし", "judgement": 1, "person_id": "tt500559", "status": 1, "temperature": 34.4 }, ...] }</pre> <p>【NG】</p> <pre>{ "result": false, "message": "An error has occurred." }</pre>

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	取得形式	type	string	"type": "1"	0: 全データ 1: 最終データ 2: date (UTC time) valid 3: id valid 4: date (local time) valid
2	アクセス日時	date	string	"date": "0000-00-00 00:00:00"	type=2 or 4 の時、有効、date 以降のアクセス履歴を取得 形式: YYYY-MM-DD HH:MM:SS
3	アクセス ID	id	string	"id": "0"	type=3 の時、有効、id 以降のアクセス履歴を取得

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	該当アクセス数	totalCount	integer	"totalCount": 27	
3	アルコール検知 結果	alco_detect	integer	"alco_detect": 0	0: 検知なし 1: アルコール測定値が設定値を超えたことを検知
4	アルコール検知器 測定結果	alco_val	double	"alco_val": 0.0	
5	自動車番号	car_num	string	"car_num": ""	
6	確認日時	confirmed_at	date	confirmed_at": "2023-06-21 8:46:49"	形式: YYYY-MM-DD HH:MM:SS
7	確認者	confirmor	string	"confirmor": "山田 拓美"	
8	データ登録日時	created_at	date	"created_at": "2023-06-21 8:39:10"	形式: YYYY-MM-DD HH:MM:SS
9	運転者	driver	string	"driver": "日立 太郎"	
10	熱検知結果	fever_detect	integer	"fever_detect": 0	0: 検知なし 1: 熱測定値が設定値を超えたことを検知
11	登録データ ID	id	integer	"id": 1	
12	指示事項	instruction	string	instruction": "なし"	
13	運転可否判断	judgement	integer	"judgement": 0	0: 運転 可 1: 運転 不可
14	ユーザ ID	person_id	string	"person_id": "tt500559"	
15	運転前後情報	status	integer	"status": 1	0: 運転前 1: 運転後
16	温度	temperature	double	"temperature": 32.0	
17	エラーメッセージ	message	string	"An error has occurred."	type, date, id のいずれかに 不具合ありの場合。もしくは例 外発生時。

3.15. 注文受付情報取得

URL	Method	Request	Response
/api/orderreceptionlog	POST	Content-Type: application/json; charset=UTF-8	
		<pre>{ "type": "0", "date": "0000-00-00 00:00:00", "id": "0" }</pre>	<p>【OK】</p> <pre>{ "result": true, "totalCount": 27, "message": [{ "confirm": 0, "created_at": "2023-06-22 13:28:12", "division1": "3333", "division2": "2222", "division3": "1111", "division4": "0000", "id": 1, "name": "日立 太郎", "order": 2, "order_str": "order_2", "person_id": "tt500559", "updated_at": "2023-06-22 13:28:27" } ...] }</pre> <p>【NG】</p> <pre>{ "result": false, "message": "An error has occurred." }</pre>

データ一覧 (Request)

#	項目	ラベル	データ型	例	説明
1	取得形式	type	string	"type": "1"	0: 全データ 1: 最終データ 2: date (UTC time) valid 3: id valid 4: date (local time) valid
2	アクセス日時	date	string	"date": "0000-00-00 00:00:00"	type=2 or 4 の時、有効、date 以降のアクセス履歴を取得 形式: YYYY-MM-DD HH:MM:SS
3	アクセス ID	id	string	"id": "0"	type=3 の時、有効、id 以降のアクセス履歴を取得

データ一覧 (Response)

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	
2	該当アクセス数	totalCount	integer	"totalCount": 27	
3	確認状態	confirm	integer	"confirm": 0	0: 確認 未 1: 確認 済
4	データ登録日時	created_at	date	"created_at": "2023-06-22 13:28:12"	形式: YYYY-MM-DD HH:MM:SS
5	組織 1	division1	string	"division1": "3333"	
6	組織 2	division2	string	"division2": "2222"	
7	組織 3	division3	string	"division3": "1111"	
8	組織 4	division4	string	"division4": "0000"	
9	登録データ ID	id	integer	"id": 1	
10	ユーザ名	name	string	"name": "日立 太郎"	
11	注文番号	order	integer	"order": 2	注文番号 1～6
12	注文タイトル	order_str	string	"order_str": "order_2"	注文番号に対応した登録文字
13	ユーザ ID	person_id	string	"person_id": "tt500559"	
14	データ更新日時	updated_at	date	"updated_at": "2023-06-22 13:28:27"	形式: YYYY-MM-DD HH:MM:SS
15	エラーメッセージ	message	string	"An error has occurred."	type, date, id のいずれかに 不具合ありの場合。もしくは例 外発生時。

4. API 要求仕様（クライアント PC 側での実装必要）

4.1. アクセス履歴最終番号更新メッセージ送信

URL	Method	Request	Response
{host_ip}/api/message/accesslog	POST	Content-Type: application/json; charset=UTF-8	
		{ "terminal_sn": "SCOPA8A27E", "id": 153 }	【OK】 { "result": true, "message": { } }

- ・メッセージは、顔認証や QR コード認証の直後に送信されます。

データ一覧（Request）

#	項目	ラベル	データ型	例	説明
1	端末 #	terminal_sn	string	"terminal_sn": "SCOPA8A27E"	
2	(最終)アクセス ID	id	integer	"id": 153	

データ一覧（Response）

#	項目	ラベル	データ型	例	説明
1	結果	result	boolean	"result": true	

5. Appendix

5.1. HTTP Header

パラメータ	値	説明
Content-Type	application/json; charset=UTF-8	
	multipart/form-data	
Cookie	ktor_session_cookie=<SessionValue>	ログインでは不要

- 取得した Cookie は 1 時間有効です。
- Cookie が無効化した場合、Header のレスポンスは HTTP/1.1 302 Found となります。
- HTTP Header のレスポンスの詳細は Curl の -v option で取得可能です。

5.2. Cookie 要否一覧

#	項目	URL	Cookie
1	ログイン	/login	否
2	端末情報取得	/api/terminalinfo	要
3	アクセス履歴取得	/api/accesslog	要
4	アクセス履歴最終番号取得	/api/accesslog/id	要
5	アクセス履歴(サムネイル)取得	/api/{user#}/accesslog/thumbnail/{id}	要
6	勤怠情報入力履歴取得	/api/attendancelog	要
7	勤怠情報入力履歴最終番号取得	/api/attendancelog/id	要
8	ユーザ登録	/api/{user#}/user	要
9	ユーザ登録情報取得	/api/{user#}/user/{id}	要
10	ユーザ登録情報修正	/api/{user#}/user/{id}	要
11	ユーザ削除	/api/{user#}/user/{id}	要
12	複数ユーザ登録情報取得	/api/{user#}/user/list/{page#}	要
13	アクセス履歴最終番号更新メッセージ要求	/api/message/accesslog	要

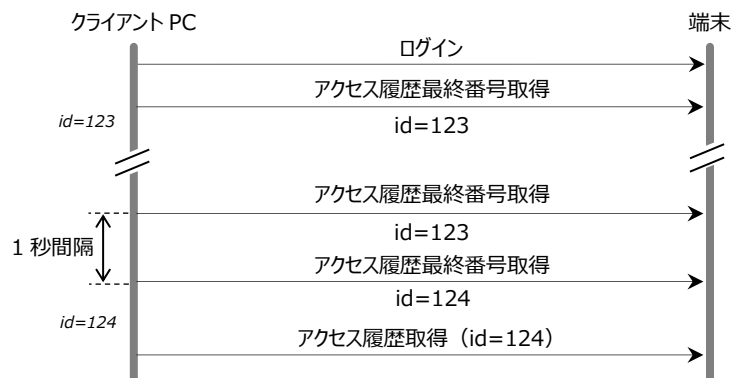
5.3. WEB API 使用例

```
curl -v -X POST http://{SERVER_IP}:8080/login
-H 'Content-Type: application/json; charset=UTF-8'
-d '{
  "username": "admin",
  "password": "aaaa"
}'
...
< Set-Cookie: ktor_session_cookie=<SessionValue>; Max-Age=3600; Expires=Thu, 27 Jan
2022 08:24:38 GMT; Path=/; HttpOnly; $x-enc=URI_ENCODING
```

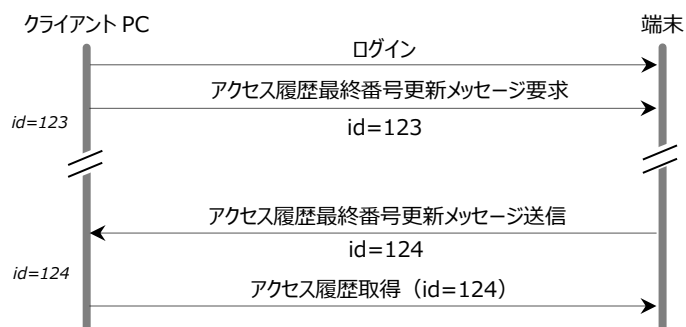
```
curl -v -X GET http://{SERVER_IP}:8080/api/terminalinfo
-H 'Content-Type: application/json; charset=UTF-8'
-b 'ktor_session_cookie=<SessionValue>'
```

5.4. 代表的なシーケンス図

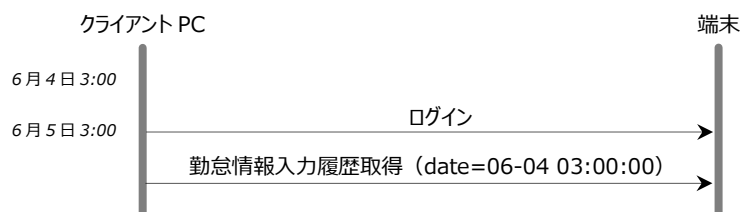
5.4.1. ユーザ情報の取得①（リアルタイム/1 秒周期）



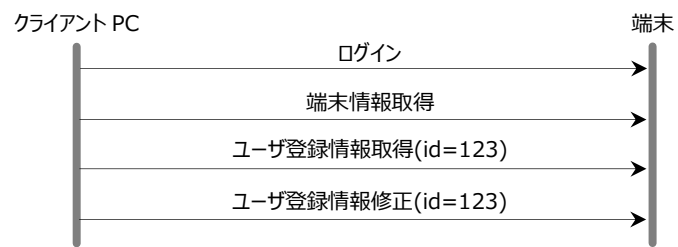
5.4.2. ユーザ情報の取得②（リアルタイム）



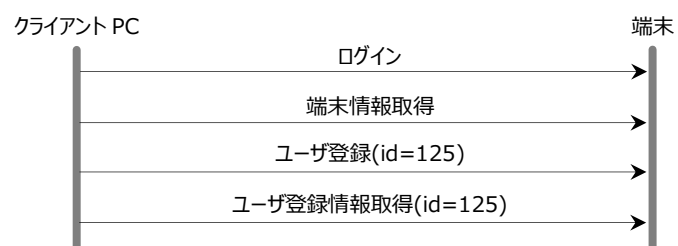
5.4.3. 勤怠情報の取得（1 回/日）



5.4.4. ユーザ情報の更新



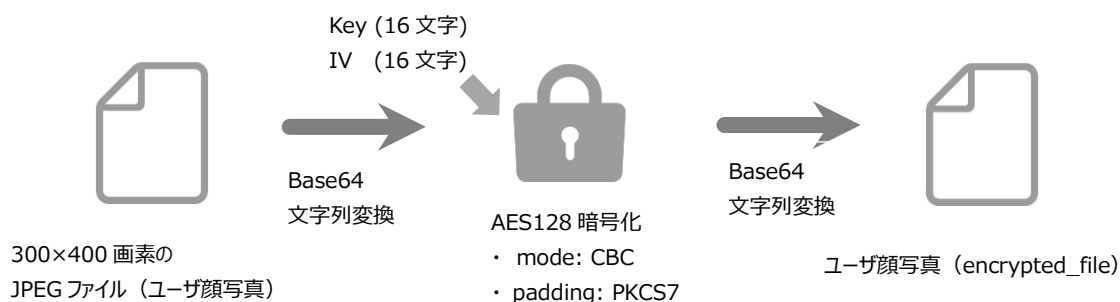
5.4.5. ユーザ情報の取得 (QRコード初期値)



5.5. サムネイルの複号化/暗号化方法

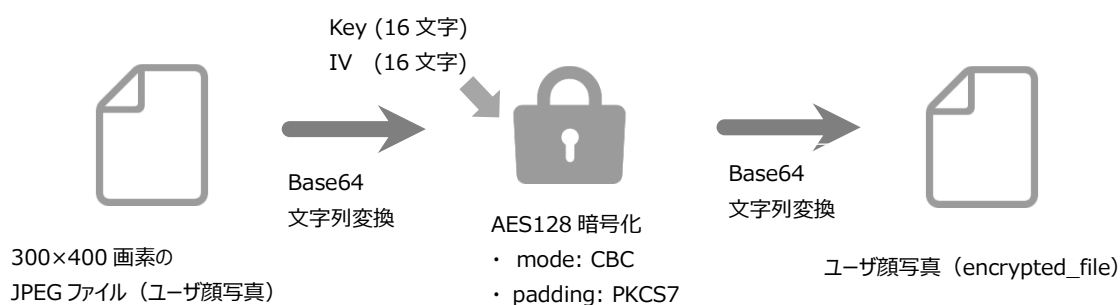
5.5.1. サムネイル” encrypted_file”の復号方法

- NDA を締結後、割り振られた AES128 用の Key と初期化ベクトル（IV）を配布します。
- 本 Key と IV を用いて、暗号化され、アクセス履歴に登録されたユーザのサムネイル（112×112 画素の JPEG ファイル）を以下の手順で AES128 にて復号してください。



5.5.2. 顔写真” encrypted_file”の暗号化方法

- NDA を締結後、割り振られた AES128 用の Key と初期化ベクトル（IV）を配布します。
- 本 Key と IV を用いて、登録するユーザの顔写真（300×400 画素の JPEG ファイル）を以下の手順で AES128 にて暗号化してください。



- JavaScript 等で、画像を Base64 形式の URL に変換して文字列として取得する場合、暗号化前に先頭の「data:image/jpeg;base64,」の部分は削除してください。

[Example]

【JavaScript 等で取得した画像の文字列】

data:image/jpeg;base64,/9j/4AAQSkZJRgAB...(中略)...999tSP8AdtHt9e9QbP8AaP5UAf/Z

【該当部分を削除した画像の文字列】

/9j/4AAQSkZJRgAB...(中略)...999tSP8AdtHt9e9QbP8AaP5UAf/Z