\Orchestrating a brighter world NEC

モバイルバックエンド基盤
REST API リファレンス
(API Gateway/Cloud Functions 編)

Ver 6.5.0

2017 年 9 月 22 日

日本電気株式会社

改版履歴

版数	日付	改版内容
5.0.0 draft1	2016/5/13	● 新規作成
5.0.0 draft2	2016/5/23	一括登録に対応. バインディングを廃止. ファンクションを新設. コードを新設. 環境を新設.
5.0.0 draft3		
5.0.0 draft4	2016/7/14	・レスポンスの内容を空から{"result":"ok"}に変更 ・10.1 カスタムロジックログ実行ログ取得 results の内容を修正 ・7.2. カスタム API の一括登録 HTTP ヘッダの application/x-yaml の記載を削除 ・下記 API の 400 Bad Request Error を削除(リクエストパラメータ、ボディの指定ができないため) 7.3. カスタム API 取得 7.4. カスタム API 一覧取得 7.5. カスタム API の削除 7.6. カスタム API の全削除 9.3. ファンクションの取得 9.4. ファンクション一覧の取得 9.5. ファンクションの削除 9.6. ファンクションの全削除 ・7.6. カスタム API の全削除 404 Not Found Error (バケットが存在しない) を追記 ・6.1 カスタム API 呼び出し 404 Not Found Error (api や function が存在しない) を追記 ・全 API 注意事項の「TBD」の記載を削除
5.0.0 draft5	2016/7/26	API 定義に ACL 指定を追記
5.0.0	2016/10/04	・10.2 システムログ取得 レスポンス内の time の値をミリ秒まで追記
6.0.0	2016/11/11	・10.1 Cloud Functions ログ取得パスを変更。
6.2.0	2017/4/19	● API 定義/Function 定義を YAML 形式でも登録・取得できるように変更
6.5.0	2017/9/22	・10.1.3. 開始日時 (start)、終了日時(end) 例) の指定誤りを修正

目次

1. はじめに

本文書は、モバイルバックエンド基盤 REST API (API Gateway/Cloud Functions 編)のリファレンスである。

本文書では API Gateway/Cloud Functions 関連各機能毎の REST API の具体的な仕様について定める。

2. 表記について

表記については、REST API リファレンス 本編を参照のこと。

3. 認証

認証については、REST API リファレンス 本編を参照のこと。

4. 共通定義

共通定義全般については、REST API リファレンス 本編を参照のこと。

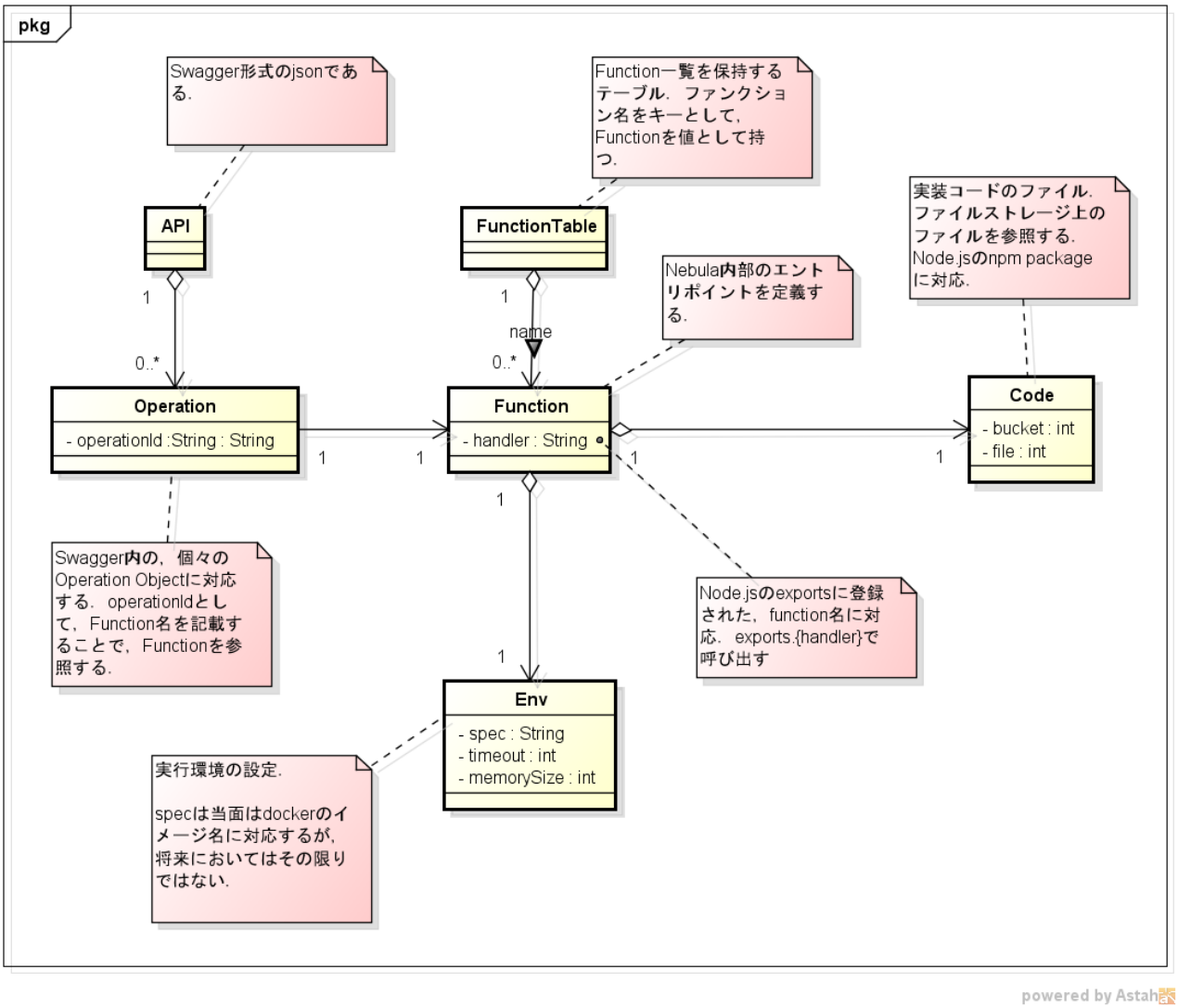
4.1. 用語定義

用語	定義
API	Nebula 外部に公開する，処理のエントリポイントの一覧．Swagger 形式で定義する．0 個以上のオペレーションを含む．
オペレーション(operation)	API を構成する要素．operationId 属性を持つ．Cloud Functions を使用する場合、operationId にファンクション名を指定し，ファンクションと関連付ける．
ファンクションテーブル(function table)	Nebula 内部で定義される，処理のエントリポイントの一覧．Nebula 独自の，ファンクションテーブル JSON で定義する．0 個以上のファンクションを含む．
ファンクション(function)	ファンクションテーブルを構成する要素．コード属性・ハンドラ属性・環境属性を持つ．
コード(code)	ファンクションを構成する要素．処理の実装コードのファイルを指定する．bucket 属性と file 属性を持つ． 処理の実装コードは，Nebula ファイルストレージに配置され，上記の bucket 属性・file 属性で参照する． 処理の実装コードファイルは，1 個以上のハンドラ実装を含む． コードの具体的な実体は，実装言語によって異なる．Node.js の場合は，npm パッケージが実体である．

ハンドラ(handler)	ファンクションを構成する要素. 文字列である. コード内のハンドラ実装を指定する. ハンドラ文字列のフォーマットは, 実装言語によって異なる. Node.js の場合は, exports.{function-name}の function-name である.
環境(env)	ファンクションを構成する要素. ファンクションを実行する環境を指定する. spec 属性, timeout 属性, memorySize 属性を含む. (spec は実質的に docker のイメージ名である)

5. データ構造

5.1. 概要



5.2. API

Swagger 形式の json で定義する.

5.3. Operation

Swagger 内の Operation Object で定義する. パス名と HTTP メソッド名で特定される.

Operation は operationId 属性を持つ. ファンクションを指定する場合、operationId は必須属性である。operationId にはファンクションなどを識別する識別名を記載する.

- ファンクションを指定する場合、識別名は "function:*Function 名*" と記述する
  - "function:" は省略可能

ベンダ拡張属性として、1) トップレベル、2) Path、3) Operation のそれぞれに ACL 属性を指定できる(複数指定した場合は後者が優先)。ACL 属性は "x-acl" 属性に設定し、値には API を実行可能なユーザ ID・グループ名の一覧を配列で指定する(グループ名は先頭に "g:" プレフィクスを付ける)。

ACL 属性を省略した場合は、権限チェックは行われない。

例を以下に示す(この例は Operation に ACL を指定している)。

```
paths:
  /sayHello:
    get:
      operationId: function:sayHello
      x-acl:
        - g:authenticated
```

5.4. ファンクションテーブル

Nebula 独自形式のファンクションテーブル JSON で定義する. キーがファンクション名, 値がファンクション JSON である.

```
{
  "{function-name-1}": { /* Function */ },
  "{function-name-2}": { /* Function */ },
  …
}
```

5.5. ファンクション

Nebula 独自形式のファンクション JSON で定義する. code 属性, handler 属性, env 属性を持つ.

```
{
  "code": { /* Code */ },
  "handler": "{handler-spec-string}",
  "env": { /* Env */ }
}
```

5.6. コード

Nebula 独自形式のコード JSON で定義する. bucket 属性, file 属性を持つ.

```
{
  "bucket": "{nebula-filestorage-bucket-name}",
  "file": "{file-name}"
}
```

5.7. 環境

Nebula 独自形式の環境 JSON で定義する. spec 属性, timeout 属性, memorySize 属性を持つ.

```
{
  "spec": "env-spec-string",
  "timeout": 600 /* seconds */,
  "memorySize": 128 /* mebi bytes */
}
```

6. APIゲートウェイ

6.1. カスタム API 呼び出し

説明	カスタム API を呼び出す
メソッド	GET/PUT/POST/DELETE (API 定義による)
パス	/1/{tenant_id}/api/{api-name}[/subpath]
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: API 定義による
リクエストパラメータ	API 定義による。指定した場合はパラメータを JSON 化(パラメータ名をプロパティ名、値をプロパティの値とする)して後続のバックエンドシステム(Cloud Functions など)に引き渡す。POST/PUT メソッドの場合は無視される。
リクエストボディ	API 定義による。指定できるのは POST/PUT メソッドの場合のみ。
レスポンスコード	● 200 OK : 正常に API を実行した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found Error : api や function が存在しない ● 500 Internal Server Error: その他エラー ● 504 Gateway Timeout : タイムアウトエラー ● その他 : API 定義による
レスポンス	API 定義による
注意事項	● subpath の有無は, API 定義による

7. カスタムAPI管理

7.1. カスタムAPI登録

説明	カスタム API を登録する
メソッド	PUT
パス	/1/{tenant_id}/apigw/apis/{api-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須) ● Content-Type: application/json, text/plain, text/x-yaml のいずれか
リクエストパラメータ	無し
リクエストボディ	API 定義(Swagger 形式 JSON または YAML)
レスポンスコード	● 200 OK : 正常に API を登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 415 Unsupported Media Type: Content-Type 不正 ● 504 Gateway Timeout : タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは

	"application/json" となる。
注意事項	● 指定した api-name で，既に API が登録されていた場合，上書きする. ● api-name は任意のパス文字列とし，クライアントが指定する

7.2. カスタムAPIの一括登録

説明	カスタム API を一括登録する
メソッド	PUT
パス	/1/{tenant_id}/apigw/apis
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須) ● Content-Type: application/json
リクエストパラメータ	無し

リクエストボディ	Nebula 独自形式の API テーブル JSON.
	
	{
	"{api-name-1}": { /* swagger 形式の json */ },
	"{api-name-2}": { /* swagger 形式の json */ },
	…
	}

レスポンスコード	● 200 OK： 正常に API を一括登録した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 415 Unsupported Media Type: Content-Type 不正 ● 504 Gateway Timeout： タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	● 指定した api-name で，既に API が登録されていた場合，上書きする. ● api-name は任意のパス文字列とし，クライアントが指定する

7.3. カスタムAPI取得

説明	カスタム API を取得する
メソッド	GET
パス	/1/{tenant_id}/apigw/apis/{api-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	● format: "json" または "text"。デフォルトは "json"。
リクエストボディ	無し
レスポンスコード	● 200 OK： 正常に API を取得した ● 401 Unauthorized： 認証エラー

	● 403 Forbidden： 権限エラー ● 404 Not Found ● 504 Gateway Timeout： タイムアウトエラー
レスポンス	リクエスト成功時は API 定義が返却される。 リクエストパラメータ format に "text" が指定された場合、登録時の定義テキスト (JSON または YAML) がそのまま返却される(Content-Type は "text/plain")。"json" を指定した場合は JSON で返却される (Content-Type は "application/json")。 エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。
注意事項	

7.4. カスタムAPI一覧取得

説明	カスタム API 一覧を取得する
メソッド	GET
パス	/1/{tenant_id}/apigw/apis/
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK： 正常に API を取得した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 504 Gateway Timeout： タイムアウトエラー

レスポンス	リクエスト成功時は API テーブル JSON.
	
	{
	"{api-name-1}": {/* swagger 形式の json */},
	"{api-name-2}": {/* swagger 形式の json */},
	…
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	

7.5. カスタムAPIの削除

説明	カスタム API を削除する
メソッド	DELETE
パス	/1/{tenant_id}/apigw/apis/{api-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK： 正常に API を削除した

	● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： 該当の API なし ● 504 Gateway Timeout： タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	

7.6. カスタムAPIの全削除

説明	カスタム API を全て削除する
メソッド	DELETE
パス	/1/{tenant_id}/apigw/apis/
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK： 正常に API を削除した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： バケットが存在しない ● 504 Gateway Timeout： タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	

8. コード管理

コードは全て, ファイルストレージに置く.

9. ファンクション管理

9.1. ファンクションの登録

説明	ファンクションを登録する
メソッド	PUT
パス	/1/{tenant_id}/functions/{function-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須) ● Content-Type: "application/json", "text/plain", "text/x-yaml" のいずれか
リクエストパラメータ	無し

リクエストボディ	BaaS 独自形式のファンクション定義。JSON または YAML で指定する。
	
	{
	"code": {
	"bucket": "myCodes",
	"file": "myCode.tar.gz"
	},
	"handler": "myJavascriptFunctionName",
	"env": {
	"spec": "node-js-6.0",
	"timeout": 300,
	"memorySize": 128
	}
	}

レスポンスコード	● 200 OK : 正常にファンクションを登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 415 Unsupported Media Type: Content-Type 不正 ● 504 Gateway Timeout : タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	

9.2. ファンクションテーブルの登録

説明	ファンクションテーブルを登録する
メソッド	PUT
パス	/1/{tenant_id}/functions
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須) ● Content-Type: "application/json", "text/plain", "text/x-yaml" のいずれか。
リクエストパラメータ	無し
リクエストボディ	BaaS 独自形式のファンクションテーブル定義。JSON または YAML。 キーにファンクション名，バリューにファンクション定義オブジェクトを指定する. {

	<pre>"foo": {
	  "code": {
	  "bucket": "myCodes",
	  "file": "myCode.tar.gz"
	  },
	  "handler": "myJavascriptFunctionName",
	  "env": {
	  "spec": "node-js-6.0",
	  "timeout": 300,
	  "memorySize": 128
	  }
	},
	"bar": {
	  …
	}
	}</pre>

レスポンスコード	● 200 OK： 正常にファンクションを一括登録した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 415 Unsupported Media Type: Content-Type 不正 ● 504 Gateway Timeout： タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	● ファンクションテーブル登録前にテナントに登録されていたファンクションはすべて削除される

9.3. ファンクションの取得

説明	ファンクションを取得する
メソッド	GET
パス	/1/{tenant_id}/functions/{function-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	● format: "json" または "text"。デフォルトは "json"。
リクエストボディ	無し
レスポンスコード	● 200 OK： 正常にファンクションを取得した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found ● 504 Gateway Timeout： タイムアウトエラー
レスポンス	リクエスト成功時はファンクション定義が返却される。 format に "text" を指定した場合は登録時のテキストがそのまま返却される (Content-Type は "text/plain")。 format に "json" を指定した場合は JSON 形式で返却される (Content-Type は "application/json")

	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。
注意事項	

9.4. ファンクション一覧の取得

説明	ファンクション一覧を取得する
メソッド	GET
パス	/1/{tenant_id}/functions
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK : 正常にファンクション一覧を取得した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found ● 504 Gateway Timeout : タイムアウトエラー
レスポンス	リクエスト成功時はファンクションテーブル定義 JSON. ファンクションテーブル定義については 9.2.「ファンクションテーブルの登録」を参照。 登録時のフォーマットに関わらず、常に JSON 形式に変換して返却する。 エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。
注意事項	

9.5. ファンクションの削除

説明	ファンクションを削除する
メソッド	DELETE
パス	/1/{tenant_id}/functions/{function-name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK : 正常にファンクションを削除した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当のファンクションなし ● 504 Gateway Timeout : タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは

	"application/json" となる。
注意事項	

9.6. ファンクションの全削除

説明	ファンクションを全て削除する
メソッド	DELETE
パス	/1/{tenant_id}/functions
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	無し
リクエストボディ	無し
レスポンスコード	● 200 OK : 正常にファンクションを全て削除した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 504 Gateway Timeout : タイムアウトエラー

レスポンス	リクエスト成功時は以下。
	{
	"result":"ok"
	}
	
	エラー時は JSON 形式でエラー通知される。Content-Type ヘッダは "application/json" となる。

注意事項	

10. ログ管理

10.1. Cloud Functions実行ログ取得

説明	Cloud Functions の実行ログを取得する
メソッド	GET
パス	/1/{tenant_id}/logs/cloudfn /1/{tenant_id}/logs/customlogic (deprecated)
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	● where : 検索条件 (オプション) ● limit : 検索数上限。デフォルト値は 100 件。(オプション) ● start : 開始日時(ISODate 形式) (オプション) ● end : 終了日時(ISODate 形式) (オプション)
リクエストボディ	無し
レスポンスコード	● 200 OK : 正常にログを取得した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 500 InternalServerError : 検索条件の演算子の利用方法が正しくない。その他のエラー。
レスポンス	条件に一致するログ情報を含む JSON データ。
注意事項	● マスターキーが必要。

	● デフォルトの検索上限数は 100 件。 ● time キー昇順でソートした結果が返却される。

Cloud Functions 実行ログをクエリする。結果は以下のように "results" に配列形式で格納される。

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "tenantId":"…",
      "appId":"…",
      "userId":"…",
      "level":"DEBUG",
      "handlerName":"testHandler",
      "functionName":"function",
      "log":"this is log",
      "time": "2013-08-27T05:19:16.000Z"
    },
    …
  ]
}
```

10.1.1. 検索条件 (where)

条件指定は where パラメータで指定する。where パラメータには、JSON で検索条件を設定する。

特定のキーに対して完全一致させたい場合は、以下のように指定する。

```
where={"handlerName": "Foo", "functionName": "Bar"}
```

その他、以下の演算子を使用できる。これらは MongoDB の演算子と同じものがそのまま使用できる。

- $lt, $gt：Less Than / Greater Than
- $lte, $gte：Less or Equal / Greater of Equal
- $ne：Not equal to
- $in
- $all
- $regex
- $exists
- $not
- $or
- $and

10.1.2. 検索数上限 (limit)

返却する検索数の上限を指定する。

以下の例では、50 件を取得する。

```
limit=50
```

limit のデフォルト値は 100 件とする。limit を指定しなかった場合は、デフォルトでこの値が指定されたもの

とみなされる。

limit に -1 を指定した場合は制限なし(無限大)として扱う。

注: サーバ側のコンフィグレーションによっては、limit 値に制限がかけられている場合がある。この場合、上限値を越える値を指定したり -1 を指定した場合は 400 Bad Request エラーとなる。

10.1.3. 開始日時 (start)、終了日時(end)

ログの開始日時と終了日時を UTC 時刻で指定する。

```
例) 2016/04/01 から 2016/05/31 までのログを取得したい場合
start=2016-04-01T00:00:00.000Z&end=2016-05-31T00:00:00.000Z
```

10.2. システムログ取得

説明	Nebula サーバ群のシステムログを取得する
メソッド	GET
パス	/1/_systemlog
HTTP ヘッダ	● X-Application-Key: システムキー (必須)
リクエストパラメータ	● where : 検索条件 (オプション) ● limit : 検索数上限。デフォルト値は 100 件。(オプション) ● start : 開始日時(ISODate 形式) (オプション) ● end : 終了日時(ISODate 形式) (オプション)
リクエストボディ	無し
レスポンスコード	● 200 OK : 正常にログを取得した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 500 InternalServerError : 検索条件の演算子の利用方法が正しくない。その他のエラー。
レスポンス	条件に一致するログ情報を含む JSON データ。
注意事項	● システムキーが必要。 ● デフォルトの検索上限数は 100 件。 ● time キー昇順でソートした結果が返却される。

Nebula サーバ群のシステムログをクエリする。結果は以下のように "results" に配列形式で格納される。

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "remoteIp":"127.0.0.1",
      "level":"INFO",
      "logger":"com.nec.baas.api.UserResource",
      "tenantId":"…",
      "appId":"…",
      "thread":"http-nio-8080-exec-5",
      "message":"Login succeeded: …",
      "time": "2016-08-27T05:19:16.000Z",
    },
```

```
    …
  ]
}
```

10.2.1. 各リクエストパラメータ

上記「Cloud Functions 実行ログ取得」の章を参照。

11. 備考

11.1. AWS lambdaとの差異

ファンクションのバージョン管理を行わない．そのため，カスタム API 管理，ファンクション管理共に，冪等な upsert に基づいた管理 API となっている．バージョン管理が必要な場合，ユーザが行うこと．API 名・ファンクション名・コード名のネーミングコンベンションに基づいてバージョン管理を行うのが望ましい．