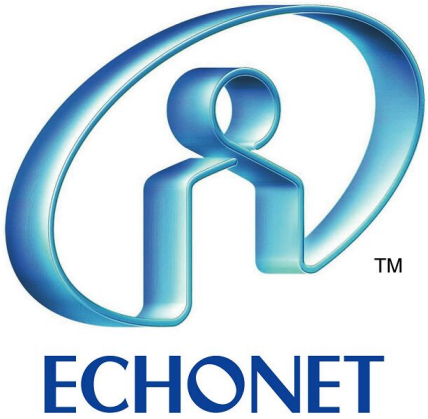
ECHONET Lite Web API
ガイドライン
API仕様部

Version 1.1.7



改定履歴

日付	版	説明
2018/10/03	Ver.1.00	制定、一般公開
2020/03/27	Ver.1.1.0 draft	バージョニング方針変更：1.**->1.*.*
		本書を「API仕様部」と名称変更
		ガイドライン構成（3章）、応用サービス（7章）追加
		旧3.7「対象ユースケースの絞り込み」を削除
		5.10に「WebhookによるINF実現例」を追加
		4章のユースケース内容を再整理
		6.5に「PATCH」メソッドを追加
		6.4 EPC/EDT/Action関連の説明追記
		表5-1に基本モデルAPI一覧を追記
		表6-3に機器オブジェクト操作API一覧を追記
		全体的に「ユーザ」「クライアント」の表現見直し
2020/08/27	Ver.1.1.0	5.7にAction Object説明追記
		6.5のSetGet例をSet例へ変更
		全体的に「リソース」「サービス」の表現見直し
		5.5にサービス例として`bulks`, `histories`を追加
2020/10/19	Ver.1.1.1	7.2のgroup description内容を修正（誤記対応）
2021/06/25	Ver.1.1.2	5.7, 7.5.2の `querySchema`を `urlParameters`へ変更
2021/07/30	Ver.1.1.3	表5-4 `urlParameter`s誤記対応
		7.1 `abort`の説明追加
		7.2 group description 不要な`descriptions`削除
		7.2 `responses[].status`の`Type`修正
		7.3 history description 不要な`descriptions`削除
		`multipleOf`, `id`/`deviceId` 誤記対応
2022/05/27	Ver.1.1.4	5.11 認証認可の内容を刷新
		6.5 内「`POST /devices/<device id>/echoCommands`」を削除。 ノード対応版へ仕様変更し7.7として追記
		7.1, 7.2, 7.3 bulk/group/historyの各descriptionにて`null`廃止
		7.1 図、例における`progress`キーの値の誤記修正

日付	版	説明
		7.2 サーバ事前登録機能追加
		7.6 定義済み機器の組み合わせ使用に関する指針追加
2023/04/28	Ver.1.1.5	7.4 にresHistoriesを追加挿入
		表6-5の200,201,204の説明文刷新
		bulk, group, history の各descriptionに関する定義表を各properties/actionsの定義を参照する形へ変更
		1.2 の参照先を更新
		全体：RFC3339のみの記載箇所にISO 8601を併記
		7章のbulk description, group description, history descriptionのschemaにおける複数型選択可能な箇所の記述変更
		7.3 historiesのretrieveアクションのリクエスト（表）におけるpageの説明を修正
2023/06/30	Ver.1.1.6	2章 対象範囲に関する図2-1 および説明を明確化
		5.4 APIバージョン一覧取得の手順にて会員IDを扱う処理を追加
		5.9、5.11参照RFC番号の刷新
2024/07/26	Ver.1.1.7	7.3 history description 不要なdescriptions削除
		5.4 infoFromServerを追加
		6.5 propertiesのキャッシュの扱い、複数プロパティリソース指定方式を追記
		7.5 に groupHistories を追加
		7章のbulk description, group description, history description, resource history description のschemaにおける複数型選択可能な箇所の記述変更

目次

図目次

表目次

第1章 はじめに

本書は、ECHONET Lite規格を活用しWeb APIによるサービス提供を実現するために考慮すべき観点や参考事例についてまとめたガイドラインとなる。ECHONET Liteのモデルを、サーバ環境を介してクライアント等へ提供し、提携サービスや応用アプリケーションの開発など利用促進に繋げるべく、Web APIを用いたシステム構築やアプリケーション実装の際に参考にすべき指針について整理する。図 1-1は、本ガイドラインにて想定する全体モデルであり、各種サービス事業者（クライアント）が ECHONET Lite Web API対応クラウド（サーバ）に対して統一的なスタイルでアクセスし、 当該クラウドを介してユーザ宅内・構内に置かれたIoT機器への操作・監視などが可能になる。本書の提供により、エコーネットの更なる普及・拡大が進むことを期待する。

以降、第2章で本ガイドラインが対象とする範囲について定め、第3章でガイドライン文書の構成、第4章で検討候補となるユースケースについて整理する。第5章では、Web API化における基本方針として、リソース設計やインタフェース設計、認証・許可方式などについて、第6章では、ECHONET Lite仕様をベースにWeb APIへの対応（変換）ルールやエラー表現などについて述べる。第7章にて、応用モデルとして、より高機能なAPIを提供するための指針について言及する。

本書は設計指針を示すガイドラインの位置づけ（「API仕様部」）とし、Web API化される具体的な機器オブジェクトのデータ型や（プロパティに相当する）各種サービス定義などは、「機器仕様部」として別書にて提供する。

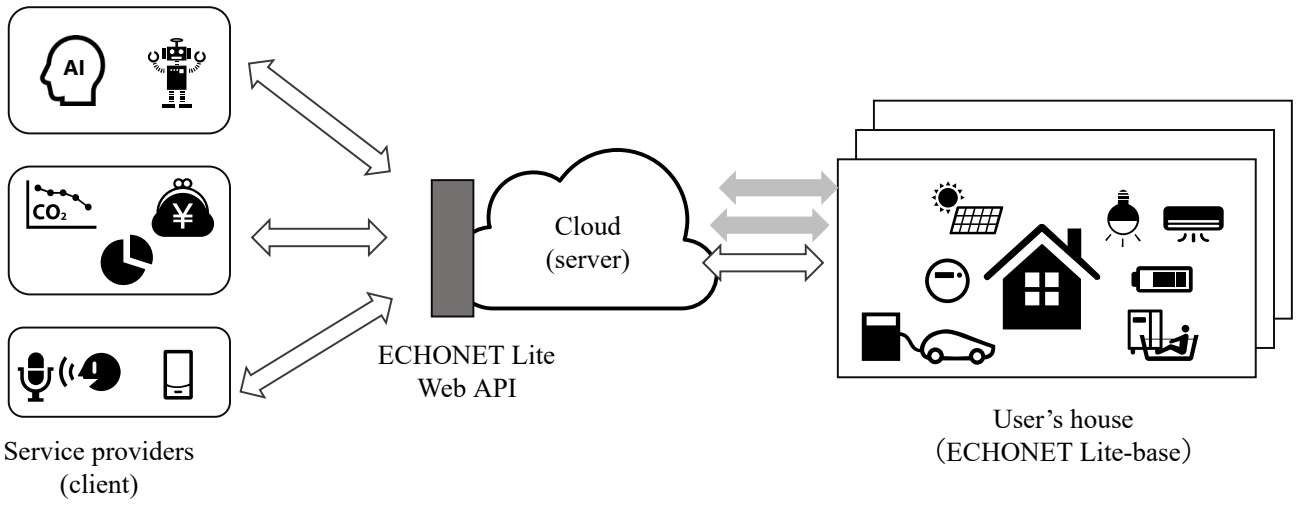


図 1-1　本ガイドラインの想定モデル

1.1 用語

用語	説明
Web API	HTTP(S)プロトコルを用いたシステム間のインタフェースであり、リクエスト／レスポンス方式でデータをやりとりする。本書では、特に呼び出される側のシステム（サーバ側）で提供されるものを指す。
RESTful	URIとHTTPメソッドを用いて操作するリソースを特定する設計原則
JSON	軽量なテキストベースの言語非依存データ交換形式

用語	説明
OAuth2.0	権限の認可(authorization)を行うためのオープンスタンダード
OpenID Connect	OAuth 2.0の拡張仕様の一つであり、ID連携をするための認証機能を定義

1.2 参照規格

本仕様で参照する規格を以下に挙げる。本仕様書に明示的な説明がない事柄については、規格文書に従う。

- [EL] The ECHONET Lite Specification Version 1.01以降
- [ELOBJ]ECHONET Specification APPENDIX: ECHONET機器オブジェクト詳細規定 Release J 以降
- [HTTP] Hypertext Transfer Protocol :
  - HTTP Semantics (RFC 9110), https://www.rfc-editor.org/info/rfc9110
  - HTTP Caching (RFC 9111), https://www.rfc-editor.org/info/rfc9111
  - HTTP/1.1 (RFC 9112), https://www.rfc-editor.org/info/rfc9112
- [JSON]The JavaScript Object Notation (JSON) Data Interchange Format (RFC 8259), https://www.rfc-editor.org/info/rfc8259
- [OAUTH] OAuth:
  - OAuth 2.0 for Native Apps (RFC 8252), https://www.rfc-editor.org/info/rfc8252
  - The OAuth 2.0 Authorization Framework: Bearer Token Usage (RFC6750), https://www.rfc-editor.org/info/rfc6750
- [OIDC] OpenID Connect Core 1.0 incorporating errata set 1,https://openid.net/specs/openid-connect-core-1_0.html

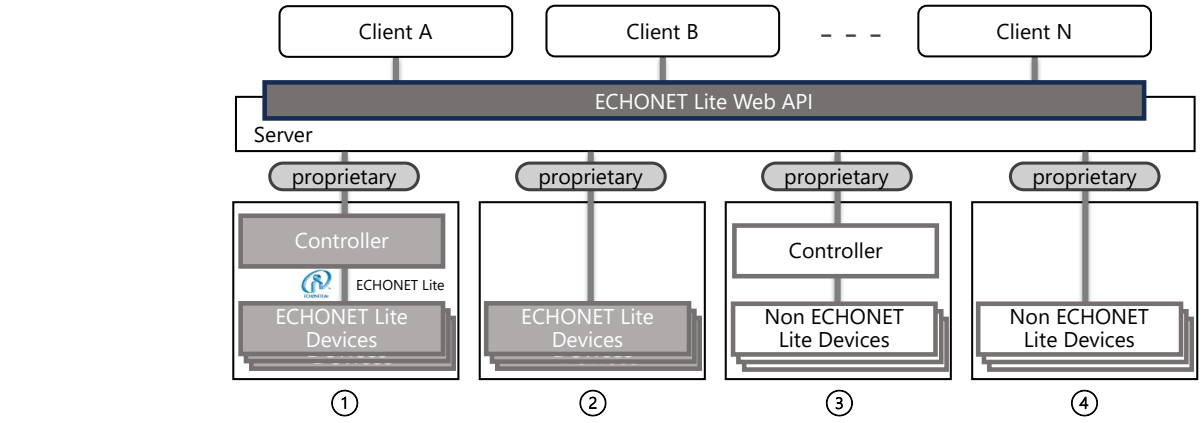
第2章 ECHONET Lite Web APIの対象範囲

本章では、対象範囲として想定するシステム構成について図 2-1に示す。

基本的には、1つの宅内に（1台以上複数可の）ECHONET Lite対応コントローラが存在し、その配下にECHONET Lite対応の機器が複数台接続されるモデルを想定する（図中①）。コントローラと機器の組み合わせは複数組存在しても良い。通常はコントローラを介してベンダ保有のサーバ（クラウド）へ接続されるケースが想定されるが、コントローラを介さず直接サーバへ接続可能な場合も対象に含んで良い（図中②）。さらに、非ECHONET Lite機器であってもサーバへマッピングすることで、本書で示すWeb API形式のモデルへ変換可能であれば、こうした機器も対象可能とする（コントローラを介する場合は図中③、介さない場合は図中④）。これらコントローラや機器をどのようにサーバ上へマッピングするかについては、実装依存（proprietary）であるとし、本ガイドラインのスコープ外とする。

また、本書では、サーバは事前にサービス・アプリ提供者であるクライアントAを知っており、適切な認証・認可を与えている（もしくは与えることが可能である）ことを前提としている。具体的な認証・認可手順は、サーバ毎の実装形態にて規定する事項とし、事例紹介に留める。

クライアントA、クライアントBなどのクライアント間でのネゴシエーションや、サーバやその他のサーバ間でのネゴシエーションも実施して構わないが、本ガイドラインでは対象としない。本書では、議論の単純化のために、クライアントとサーバ間にてサーバ側が提供するWeb APIについてのみ規定する方針とした。



Server通信プロトコル（proprietary）搭載位置	コントローラ	機器	コントローラ	機器
コントローラと機器間の通信プロトコル	ECHONET Lite	ー	独自	ー
機器搭載機能モデル	ECHONET Lite	ECHONET Lite	独自	独自

図 2-1　本ガイドライン対象の基本システム構成図

あらためて、本ガイドラインが規定するAPIの範囲を図 2-2に示す。ECHONET Lite Web APIは、サーバがクライアントに対して提示するWeb APIのみを対象としており、サーバから宅内との通信部については対象外（ベンダ固有）とする。

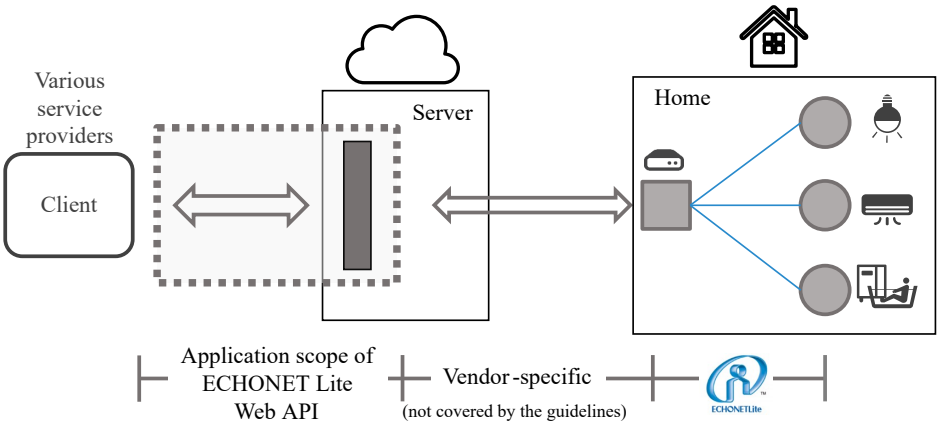


図 2-2　本ガイドラインの対象範囲

第3章 ガイドラインの構成

ECHONET Lite Web APIガイドラインは、「API仕様部」（本書）と「機器仕様部」（別書）から構成される。

「API仕様部」は、同ガイドラインが対象とするユースケースやWeb APIモデルの指針、ECHONET Lite仕様からWeb APIへのマッピング指針などが提供される。

「機器仕様部」は、主に、機器毎のDevice Description（搭載データ型などのスキーマ規定：後述）を例示したもので、使用するデータ型やネーミング指針なども提供される。

各仕様部の更新・メンテナンス方針として、API仕様部の更新は、応用サービスなどの機能拡充や仕様追加などに伴い実施され、機器仕様部の更新は、対象機器やプロパティの追加などに伴い実施されるものとしている。

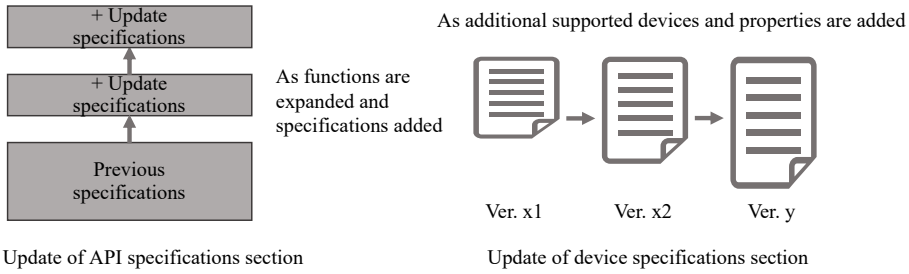


図 3-1　API仕様部・機器仕様部の更新方針

本ガイドラインのバージョニングは、セマンティックバージョニングを採用する。Version X.Y.Z（メジャーバージョン、マイナーバージョン、パッチバージョン）の形式とし、各々、API変更の際に互換性がなくなる場合はメジャーバージョンを上げ、後方互換性を保ちつつ機能を追加・変更する場合はマイナーバージョンを上げ、後方互換性を伴うバグ修正などはパッチバージョンを上げる。

第4章 ECHONET Lite Web APIのユースケース

本章では、本ガイドライン策定におけるユースケースについてまとめる。

4.1 状態取得/制御/通知

ECHONET Liteの基本操作（ESV）であるGET/SET/INFをマッピングするユースケース（図 4-1）。GETについては、宅内機器までの応答遅延を考慮し、サーバ内でキャッシングしたデータをクライアントへ返却するケースも含む。SETは、基本的に宅内機器への操作を想定する。INFのように、機器にて生じた状態変化通知などをできるだけ即座にサーバを介してクライアントへ伝えるケースも想定される。

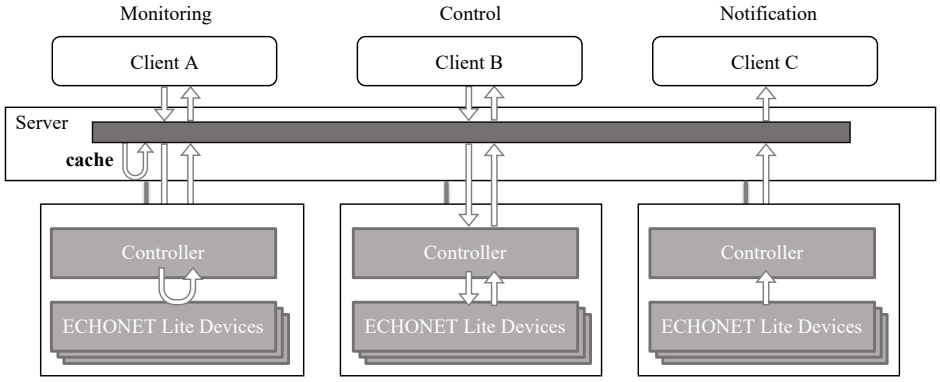


図 4-1　ユースケース：状態取得/制御/通知

4.2 機器の一覧取得/管理情報取得

機器の一覧（リスト）を取得するユースケース（図 4-2の左）。ある機器ユーザ宅の機器全ての一覧を取得するケースや、エアコンのみといった機種指定による機器の一覧を取得するケースが考えられる。さらに、サーバが収容している全ての機器ユーザが保有する機器全ての一覧や、その一部を切り取って扱うケースなども考えられる。

また、ECHONET Liteコントローラは機器オブジェクトとしてコントローラクラスを搭載しており、オプション機能として同コントローラが管理対象とする機器のリストなど、機器管理情報を保持することが可能となっている。こうした機器管理情報をサーバ側へマッピングし、クライアントから取得可能とするケースが考えられる。機器管理情報は、サーバ上でキャッシュし、機器追加・削除などの変更時のみ通知によりサーバ内にキャッシュした機器管理情報を更新するケースも想定される（図 4-2の右）。

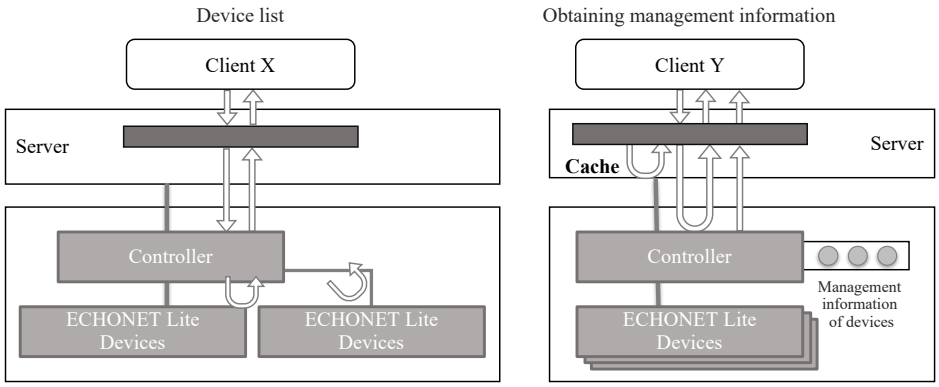


図 4-2　ユースケース：機器の一覧取得/管理情報取得

4.3 機器の一括操作

ECHONET Liteでは、コントローラ配下の全機器への一括操作や、エアコン全てといった機種指定による一括操作がサポートされている。これらをクライアントから操作可能とするユースケース。一人のユーザだけでなく、複数の機器ユーザ宅内の機器を一斉に操作するケースも考えられる。（宅内もしくは複数の機器ユーザを跨がり）複数の機器を一斉に操作する場合、同一のプロパティへの操作を行うケースもあれば、機器ごとに異なるプロパティへの操作を行うケースもある。また、操作指示（GET/SET）についても機器ごとに異なるケースも考えられる。一台の機器に対して複数の操作を行うケースもありうる。いずれも、一度の指示で一台以上の機器に対して複数の操作が行えるケースとして考えることができる。

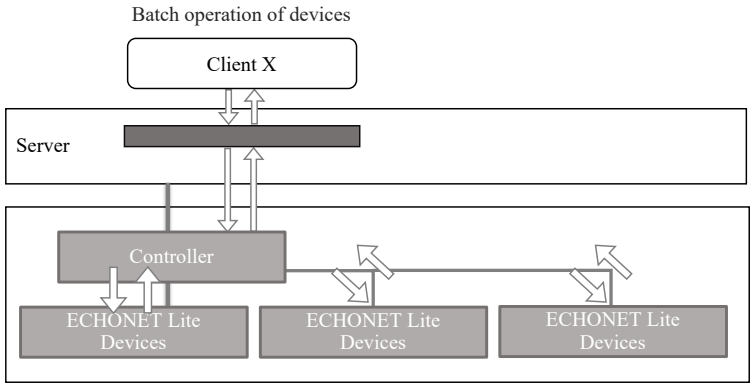


図 4-3　ユースケース：一括操作

4.4 仮想機器

宅内の機器をサーバ上へマッピングする際、様々な情報加工を施すことで、見かけ上、機器の機能を拡張することが可能なユースケース。仮想機器として見せかけるバリエーションとしてここでは3種の想定例について列挙する（図 4-4）。

例えば、ある機器オブジェクト仕様Appendix Release X対応のエアコンをサーバへマッピングする際、Release X時から最新のRelease Yにおいても規格書の該当内容に関するプロパティについて変更が無い場合は、Release Y対応エアコンとしてクライアントへ見せかけることが可能となる。旧モデル対応エアコンであっても、サーバ上での適切な変換処理により、新モデル対応エアコンと同

様に扱えるなら、新モデルにしか対応しないクライアント提供アプリにも対応が可能となるメリットがある（図左）。

また、宅内のエアコンとなんらかのセンサを組み合わせて、新センサ付仮想エアコンとみなして、クライアントのアプリへ提供することも考えられる。エアコンとセンサを個別の機器として扱っても良いが、一体化することによりアプリの幅が広がる可能性がある（図中央）。

さらに、エアコン自体が保持しない機能であっても、サーバ上で付加機能を加えることで拡張エアコンとしてクライアントへ公開するケースも考えられる。天気予報付きエアコンのようなものがあっても良いかもしれない（図右）。

仮想機器を扱う事例のバリエーションとして、非ECHONET Lite機器を扱うことも考えられる。 複数のエネルギー関連機器が保有するエネルギー関連情報（例えば、発電能力、蓄電能力、負荷情報など）を収集し、サーバ内で集約することで仮想（EMS）情報機器としてクライアントへ提供することも考えられる。 複数の機器を同一目的の下にとりまとめ、例えば、エネルギー情報の取得や、充放電といったエネルギー制御などを可能とする操作APIを規定することで、ひとつの仮想的な機器として扱うことが考えられる。

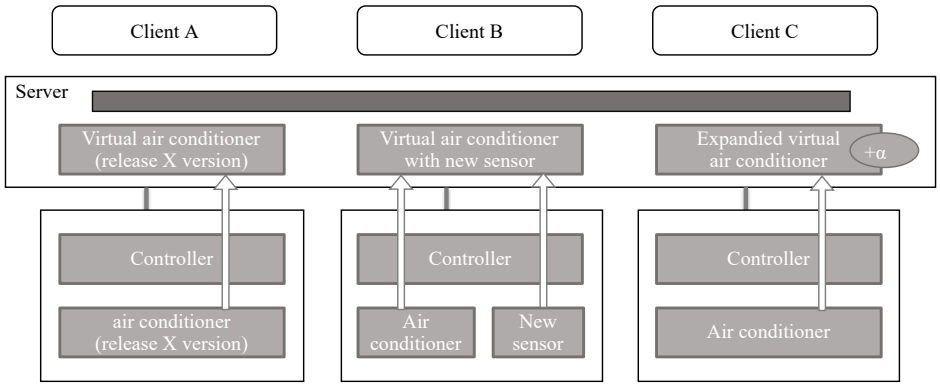


図 4-4　ユースケース：仮想機器

4.5 サーバ蓄積ログ

宅内機器やセンサから得られる情報をサーバ上にて蓄積ログとして集積し、生活行動分析などに繋げるサービスも考えられる。 これらは、直接宅内ECHONET Lite機器をマッピングしたものではなく、機器から収集される情報を集約・加工したものをクライアントへ提供する形式となる（図 4-5）。

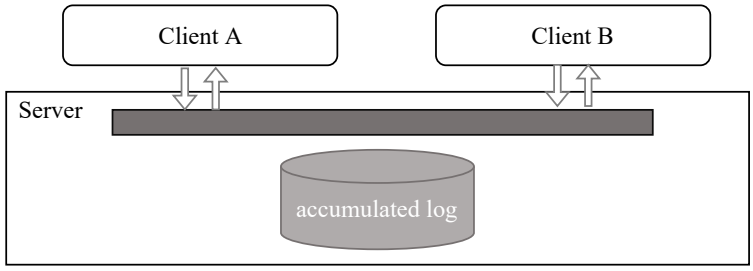


図 4-5　ユースケース：サーバ蓄積ログ

4.6 認証/認可

今回規定するような顧客情報を扱うケースでは、そもそもクライアントがサーバからWeb APIによるサービス提供を受けるにあたり、なんらかの認証や認可を得る必要があるのが一般的である。具体的には、OAuth2.0 [OAUTH]などのプロトコルを用いてセッションを確立した上で、サーバは、クライアント毎にアクセス許諾するリソース集合を準備し、実際にサービス提供することになる。

また、サーバ上では、顧客情報の追加／削除や、顧客保有のコントローラや機器の追加／削除を実施可能な仕組みが必要となる。こうした顧客管理、機器管理機能をWeb APIにて外部提供することも考えられるが、サーバと顧客間であらかじめ必要な作業を行う形式も考えられる（図 4-6）。

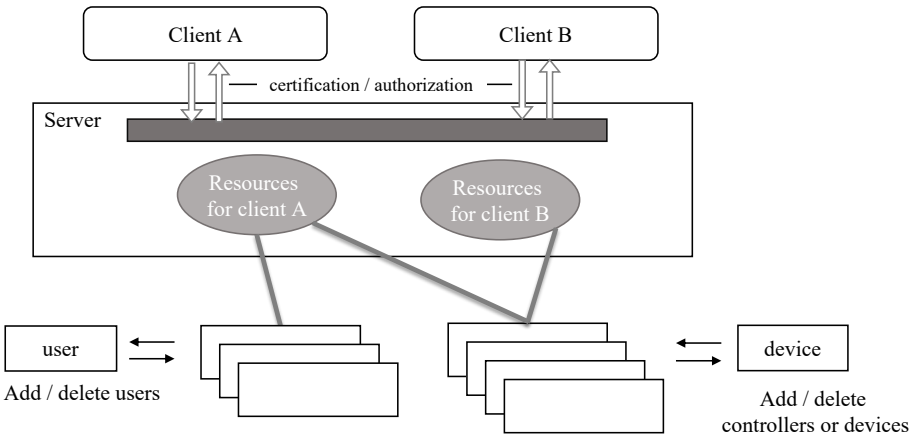


図 4-6　ユースケース：認証/認可

第5章 Web APIモデルの指針

5.1 基本方針

本ガイドラインでは、Web APIのモデルとして、基本的にREST（REpresentational State Transfer）を採用する。RESTは、URIに対してHTTP [HTTP]メソッドをベースにシンプルな制御が可能であり、現在実質的なWeb APIの標準になっている。データ形式には現在主流であるJSON [JSON]を採用し、Rest(ful) APIとなるよう全体設計する。

RESTの4原則である、①HTTPによるステートレス性、②URIによるアドレス可視性、③HTTPメソッドによる統一インタフェース、④JSONによるリソース間の接続性に留意しつつ、APIを規定する。 リソースの識別方法は、下記を基本とする。

```
形式    スキーム://ホスト名/リソースへのパス
例      https://www.example.com/elapi/v1/devices/12345678
```

この例では、HTTPSスキームにて、ホスト www.example.comが保有するIDが12345678の機器データのリソースを指定している（ポート番号については省略）。このように、パスで階層化してリソースを指定するREST APIを基本モデルとする。また、クエリーパラメータでリソースを指定するクエリーAPIも補完的に選択可能とする。具体的なリソースの指定方法については、次節以降記述する。

リソースに対するアクションの多くは、CRUD（Create、Read、Update、Delete）で表現される作成、参照、更新、削除の各操作を参考に、リソースのURIに対するHTTPメソッドやURIのクエリーパラメータにて実装されるモデルを想定する。

前述以外の基本方針として、下記を定める。

- HTTPのバージョンはHTTP/1.1[HTTP]を採用する。
- JSON形式は、RFC 8259に従う。Content-Typeとして「"application/json"」を指定し、文字コードはUTF-8を使用する。クライアント側では、Acceptというリクエストヘッダでメディアタイプ「"application/json"」を指定する。
- スキームには、HTTPSを使用する。
- 日時形式には、RFC 3339（ISO 8601）を採用し、タイムゾーンを考慮した下記表記に従う。
  - 2018-01-02T12:34:56+09:00
  - ただし、年月日、時、分、秒など個別に指定すべきケースは個々の記述に従う。
- リソースに指定されるプロパティ名称など属性名の命名規則として、単一語の場合は小文字を使用するが、複合語の場合は（ローワー）キャメルケースを使用する。本ガイドラインでは説明の都合、任意の値や記述を表現する場合には山括弧「< >」を使用する。
- ECHONET Liteの機器オブジェクトのDevice Description形式（後述）は、W3Cで策定中のWoTモデルを一部参考として記述する。

次節以降、下記のサービス指定・操作例について記述する。

- アプリケーション名
- APIバージョン、APIバージョン一覧取得
- 対象サービス種一覧取得
- 機器の一覧取得

- 機器情報（Device Description）取得
- 機器オブジェクトのプロパティ値操作（SET/GETなど）
- キャッシュの扱い
- 機器オブジェクトのプロパティ値通知（INF）
- 認証・認可

以下で示す事例では、HTTP(S)リクエストでは、リクエストライン、リクエストヘッダ、メッセージボディ、HTTP(S)レスポンスでは、ステータスライン、レスポンスヘッダ、メッセージボディの内容のうち、（リクエスト／レスポンス）ヘッダ部については特別明記する必要がある場合を除き例示を省略する。

表 5-1に、基本モデルに関するAPI一覧を示す。以降、個別に説明する。

表 5-1　本書規定の基本モデルに関するAPI

http method	path	description
GET	/elapi	APIバージョン一覧取得
GET	/elapi/v1	対象サービス種一覧取得
GET	/elapi/v1/<サービス指定省略可）>/devices	機器一覧取得
GET	/elapi/v1/devices/<device id>	機器情報（Device Description）取得（サービス種省略）

5.2 アプリケーション名

本書にて規定するECHONET Lite Web APIを用いたアプリケーション名称を「elapi」と呼称する。トップ階層に「/elapi」というアプリケーション名をパスとして埋め込むことで、ECHONET Lite Web APIのコンポーネントであることを示す。 なお、ベンダにより「elapi」とは異なる別のアプリケーション名称を使用したい場合は、ベンダ所望の名称を用いても構わない。以降、本書で示される事例では、アプリケーション名称を「elapi」と呼称するが、ベンダ所望の名称とする場合は適時読み替えること。

本書にて特定パス収容方式を選定した理由

> APIの指定方式として、下記のような記述例が候補となる。
> ①特定のパスに収容する場合　https://www.service.xx/elapi/xxx
> ②ホスト名で区別する場合　　https://elapi.service.xx/xxx
>
> Webアプリケーションの実装形態やサービス規模／負荷分散も考慮すべきだが、リバースプロキシなどを用いれば接続先Webアプリケーションサーバの切り替えは可能なため、本書では①特定パスに収容する方式に統一する。なお、上述の通り、ベンダ所望のアプリケーション名称を用いる場合は、②の形式であっても構わない。

5.3 APIバージョン

本ガイドラインで定めるAPIバージョンはリビジョン番号形式で管理するものとし、現時点ではv1とする。今後、細かな仕様追加や修正などを実施する場合であっても、後方互換性がとれる範囲で規定する場合は、バージョンを上げずに運用するものとする。将来、互換性の無い仕様変更が必要となった場合は、仕様を区別するため、v2を策定する方針とする（メジャーバージョンが1から2へ更新されたものとする）。なお、バージョン指定は、URIにバージョン情報を埋め込む方法を採用する。

なお、前述の「アプリケーション名」と同様、ベンダ固有のバージョニングを行う場合は、この規定に従う必要はない。ベンダにて規定される方針に従い対応すること。以降、本書で示される事例では、APIバージョンを「v1」とするが、ベンダ所望の形式とする場合は適時読み替えること。

URI指定によるバージョン管理を選択した理由

> バージョン指定方法には、下記のように大きく3通りの方法がある。
> ① URIに含める： https://www.service.xx/elapi/v1
> ② クエリーパラメータに含める：https://www.service.xx/elapi/xx/80234512?v=1.5
> ③ HTTPヘッダに含める：`Accept: application/vnd.echonet.v2+json`
>
> ②はバージョン情報が省略可能というメリットがあるが、デフォルトバージョンが最新版とすると、API変更時にトラブルを発生しやすい。また、③はベンダ固有のメディアタイプとしてサーバへ要求し、対応可能な場合は「`Content-Type: application/vnd.echonet.v2+json`」と共に「`Vary: Accept`」をヘッダに付与して応答する必要があるが、`Content-Type`が`application/json`と一致しないと（独自のメディアタイプを）受け付けないライブラリが存在するケースもある。 本書では、識別しやすく、多くのWeb API提供サービスで広く使われていることを理由に、①を採用する。

5.4 APIバージョン一覧取得

トップのURIに対して「`/elapi`」を指定し、GETメソッドにて要求することで、サーバが対応するバージョン一覧を取得できる。

■ リクエスト

```
GET /elapi HTTP/1.1
```

■レスポンス

```
{
    "versions": [
        {
            "id": "v1",
            "status": "CURRENT",
            "updated": "2018-01-01T12:34:56+09:00",
            "infoFromServer": {
                "apiVersion": "1.1.7",
                "ddVersion": "1.6.0",
```

```
                "supportedDeviceTypes": [
                    "homeAirConditioner",
                    "generalLighting"
                ]
            }
        },
        {
            "id": "v2",
            "status": "EXPERIMENTAL",
            "updated": "2018-01-02T01:02:03+09:00"
        }
    ]
}
```

上記例では、v1と v2のふたつのバージョンをサポートしていることを示している。最低1つのバージョンはサポートすること。本書に対応したバージョンは v1とし、将来のバージョン更新は v2を予定する。開発レベルなどに応じて、statusには、下記の状態を指定可能とする。

- CURRENT：最新の安定版
- SUPPORTED：安定版（最新版ではない）
- DEPRECATED：既に廃止となった版
- EXPERIMENTAL：開発・実験中の版

updatedで、更新日時を指定可能とし、サーバ側システムに何らかの変更を実施したタイミングを記述可能とする。

また、infoFromServerにて、必要に応じてサーバからの追加情報を自由に指定可能とする。上記では、API仕様部、機器仕様部の準拠バージョン情報や対応機器情報などを記載する例を示している。追加情報となる各項目は特に定義しないため、サーバ都合にて自由に規定して良い。

idは必須、status, updated, infoFromServerはオプション指定とする。

なお、本ガイドラインをベースとしたクライアント（アプリケーション）やサーバ（システム）を構築するにあたり、下記に示す手順は、本ガイドラインを利用している旨公開する上で望まれる項目の1つとなる。クライアントによる要求時はHTTP拡張ヘッダにて会員ID（ECHONET-member-id）を指定し、サーバからの応答時はレスポンスボディに会員IDを追加する。会員IDの値は、エコーネットコンソーシアム会員へ発行される会員IDの値（3バイト16進数）を使用すること。

■ リクエストGET

```
GET /elapi HTTP/1.1

ECHONET-member-id: "001122"
```

■ レスポンス

```
{
    "versions": [
        {
            "id": "v1",
            "status": "CURRENT",
            "updated": "2018-01-01T12:34:56+09:00",
            "infoFromServer": {
                "apiVersion": "1.1.7",
                "ddVersion": "1.6.0",
                "supportedDeviceTypes": [
                    "homeAirConditioner",
                    "generalLighting"
                ]
            }
        },
        {
            "id": "v2",
            "status": "EXPERIMENTAL",
            "updated": "2018-01-02T01:02:03+09:00"
        }
    ],
    "ECHONET-member-id": "334455"
}
```

本ガイドラインの利用を公開するクライアントはリクエスト時に上記HTTP拡張ヘッダを指定する。一方、本ガイドラインの利用を公開するサーバは、クライアントが `ECHONET-member-id`ヘッダを使用する／しないに関わらず、レスポンス時に `ECHONET-member-id`の値を指定する。

5.5 対象サービス種一覧取得

「`/elapi/<version id>`」を指定し、GETメソッドにて要求することで、サーバが「`/elapi/<version id>`」直下で対応するサービス種一覧を取得できる。

■ リクエスト

```
GET /elapi/v1 HTTP/1.1
```

■ レスポンス

```
{
    "v1": [
        {
            "name": "devices",
            "descriptions": {
```

```
                "ja": "devicesの説明文",
                "en": "device resource"
            },
            "total": 10
        },
        {
            "name": "controllers",
            "descriptions": {
                "ja": "controllersの説明文",
                "en": "controller resource"
            },
            "total": 1
        },
        {
            "name": "sites",
            "descriptions": {
                "ja": "sitesの説明文",
                "en": "sites resource"
            },
            "total": 5
        },
        {
            "name": "users",
            "descriptions": {
                "ja": "usersの説明文",
                "en": "user resource"
            },
            "total": 100
        },
        {
            "name": "groups",
            "descriptions": {
                "ja": "groupsの説明文",
                "en": "group resource"
            },
            "total": 3
        },
        {
            "name": "bulks",
            "descriptions": {
                "ja": "bulksの説明文",
                "en": "bulks resource"
            },
            "total": 10
        },
        {
            "name": "histories",
            "descriptions": {
                "ja": "historiesの説明文",
```

```
                "en": "histories resource"
            },
            "total": 20
        }
    ]
}
```

上記例では、devices, controllers, sites, users, groups, bulks, historiesの7種類のサービス種をサポートしていることを示している。本ガイドラインではこれらのうちいくつかのサービス種について例示するが、同様のサービス種をサーバが任意に規定できるものとする。サービス種は、必ず1種類以上提供すること。クライアントに対して提供しないサービス種は記載しないこと。各サービス種の説明文を descriptions（必須）にて、各オブジェクトの総数を total（オプション）で示す。説明文の記述は任意の文字列とし、総数は整数値とする。基本的に、クライアント（提供先）の要求・権限や提供サービス内容などに応じて、各クライアントへ提供するサービス種を変更して（切り替えて）構わない。

以降、本ガイドラインではdevices, bulks, groups, historiesのケースについて規定する。他のサービス種については任意に規定可能とし、本書ではスコープ外とする。

5.6 機器一覧取得

「/elapi/v1/<サービス指定（省略可）>/devices」 を指定してGETすることで機器一覧を取得できる。

■ リクエスト

```
GET /elapi/v1/devices HTTP/1.1
```

■ レスポンス

```
{
    "devices": [
        {
            "id": "0xFE000006123456789ABCDEF123456789AB",
            "deviceType": "generalLighting",
            "protocol": {
                "type": "ECHONET_Lite v1.13",
                "version": "Rel.J"
            },
            "manufacturer": {
                "code": "<manufacturer code>",
                "descriptions": {
                    "ja": "description in Japanese",
                    "en": "description in English"
                }
```

```
            }
        },
        {
            "id": "",
            ...
        }
    ]
}
```

表 5 - 2　機器一覧取得時の応答内容

Property	Type	Required	Description	Example
id	string	Yes	機器固有のID	"0xFE000006…AB"
deviceType	string	Yes	機器の種類	"generalLighting"
protocol	object	Yes	使用プロトコル	ー
protocol.type	string	Yes	ECHONET Liteバージョン番号	"ECHONET_Lite v1.13"
protocol.version	string	Yes	Appendixリリース番号	"Rel.J"
manufacturer	object	Yes	メーカ情報	ー
manufacturer.code	string	Yes	メーカコード	"0x******"
manufacturer.descriptions	object	Yes	メーカ名称	ー
manufacturer.descriptions.ja	string	Yes	名称（日本語）	"A ベンダ"
manufacturer.descriptions.en	string	Yes	名称（英語）	"Vendor Name A"

idは、機器固有のIDであり、システム内（/devices配下）にて重複しないユニークな値である必要がある。値の例としては、機器自体が保持する固有情報（MACアドレスや機器オブジェクトの固有IDなど）や、コントローラが割り当てる値、サーバ内にて固有に割り当てる値などが想定される。上記例では先頭に"0x"を付加した16進数の文字列にて記載しているが、"0x"の無い任意の文字列で構わない。

deviceTypeは、機器オブジェクトのクラス名に相当する名称となる。 ECHONET Liteの機器オブジェクトに対応付けられるクラス名称については、「機器仕様部」（別書）にて記載する。

protocolは、typeでECHONET Liteのバージョン番号を、versionで機器オブジェクトのAppendixリリース番号（EPC=0x82）をASCII文字列にて記載する。

manufacturerは、ECHONET Liteの機器オブジェクトに対応付けられる機器の場合は、機器オブジェクト・スーパークラスのメーカコード（EPC=0x8A）のプロパティ値をcodeに、descriptionにて日本語(ja)、英語(en)での名称を記載する。

表5-2のRequired列の扱いについては、機器の独自拡張を実施する場合は7.5節記載の通り、柔軟に変更可能とする。

機器一覧は多数となる場合がある。サーバが一度にクライアントへ返却する機器数を制限したい場合には、下記のように、後続機器の有無（"hasMore"）、一度に返却する数（"limit"）、先頭からのオフセット位置（"offset"）を合わせて返却しても良い。

```
{
    "devices": [
        ...
    ],
    "hasMore": true,
    "limit": 25,
    "offset": 0
}
```

また、クライアント側からオフセット位置と返却数を指定するクエリー形式もサポートしても良い。

```
    GET /elapi/v1/devices?offset=0&limit=25 HTTP/1.1
```

5.7 機器情報（Device Description）取得

「/elapi/v1/<サービス指定（省略可）>/devices/<device id>」 を指定してGETすることで、機器に対応する仕様をJSON 形式の 記述「Device Description」として取得可能とする。Device Descriptionは、機器が実装している機能（properties, actions, events）の定義である。ECHONET Liteの機器オブジェクトに対応付けられる機器の場合、各機器のDevice Descriptionは「機器仕様部」（別書）に記述されている（現在、主要機器について記述）。

■ リクエスト

```
GET /elapi/v1/devices/<device id> HTTP/1.1
```

■ レスポンス

```
{
    "deviceType":<device type>,
    "eoj":<eoj in Hex string>,
    "descriptions": {
        "ja": <description of property in Japanese>,
        "en": <description of property in English>
    },
```

```
    "properties": { <property1>: <property object>, <property2>: <property
object> ...    },
    "actions":  { <action1>: <action object>, <action2>: <action
object>...    },
    "events": { <event1>: <event object>, <event2>: <event object>...    }
}
```

表 5-3　機器情報取得時の応答内容

Property	Type	Required	Description	Example
deviceType	string	Yes	device の種類を示す。ECHONET Liteの機器オブジェクト名(EOJ)に対応する。値に関しては「機器仕様部」（別書）の「5. 機器毎のDevice Description」を参照のこと。	"generalLighting"
eoj	string	No	ECHONET LiteのEOJのクラスコードをstringでHex表記（IDとして使用しない）	"0x0130"
descriptions	object	Yes	ECHONET Liteで定義された機器オブジェクトの名称	—
descriptions.ja	string	Yes	ECHONET Liteの機器オブジェクト（日本語）	"一般照明"
descriptions.en	string	Yes	ECHONET Liteの機器オブジェクト（英語）	"General Lighting"
properties	object	Yes	property object の列挙。property1,property2は、property resource name（「機器仕様部」（別書）記載）	
actions	object	No	action object の列挙。action1, action2は、action resource name（「機器仕様部」（別書）記載）。	
events	object	No	event object の列挙。event1, event2は、event resource name。	

Property object

Property objectは機器のProperty定義を記述する。ECHONET LiteでGETまたはSETをサポートするエコーネットプロパティに対応する。本Web APIでpropertyを取得するにはGET メソッド、設定するにはPUTメソッドを使用する。property objectの構成を以下に示す。

```
{
    "epc":<epc in Hex string>,
    "epcAtomic":<epc in Hex string>,
    "descriptions": {
        "ja": <description of property in Japanese>,
        "en": <description of property in English>
    },
    "writable":<writable flag>,
    "observable":<observable flag>,
    "urlParameters": <schema of parameters>,
    "schema":<schema of data>,
    "note": {
        "ja":<note in Japanese>,
        "en":<note in English>
    }
}
```

表 5-4　Property Objectの内容

Property	Type	Required	Description	Example
epc	string	No	対応するECHONET LiteのEPCをstringでHex表記する	"0x80"
epcAtomic	string	No	ECHONET LiteとしてAtomic operationが必要なEPC	"0xCD"
descriptions	object	Yes	ECHONET Liteで定義されたエコーネットプロパティの名称	—
descriptions.ja	string	Yes	ECHONET Liteのプロパティ名（日本語）	"動作状態"
descriptions.en	string	Yes	ECHONET Liteのプロパティ名（英語）	"Operation Status"
writable	boolean	Yes	書き込みが可能か不可能かを示す。ECHONET LiteのSETに対応	true, false

Property	Type	Required	Description	Example
observable	boolean	Yes	状態変化を通知できるか否かを示す。ECHONET LiteのINFに対応	true, false
urlParameters	object	No	GETでparameterを渡す必要がある場合にqueryを利用する。parameterの情報をJSON Schema形式で記述	{ "day": { "descriptions": { "ja": "xx", "en": "yy" }, "schema": { "type": "number", "minimum": 0, "maximum": 99 }, "required": false }}
schema	object	Yes	property dataの情報をJSON Schema形式で記述	{"type": "string","format": "time"}
note	object	No	Propertyに関する付加情報	—
note.ja	string	No	付加情報を日本語で記述する	"秒の指定は無視される"
note.en	string	No	付加情報を英語で記述する	"number of seconds is ignored"

Action object

Action objectは、propertyでは表現しにくい、機器が提供する機能を記述する。例として、SETのみのproperty操作、property定義をしない機器の内部状態の変更、複数のpropertyのアトミックな変更、時間を要する機器状態の変更(例 時間をかけて照明のフェージングを行う)が挙げられる。また、機器の内部状態に無関係な純粋な関数機能(入力引数だけに対して演算した結果を出力で返す)をaction定義しても良い。ECHONET Liteの機器オブジェクト定義にactionは存在しないが、例えば、電気自動車充電器の"積算電力量リセット設定"プロパティにSETすることで積算充電電力量を0にリセットする機能を、プロパティの代わりにactionで定義することが考えられる。本Web APIでactionを実行するにはPOST メソッドを使用する。action objectの構成を以下に示す。

```
{
    "epc": <epc in Hex string>,
    "descriptions": {
        "ja": <description of action in Japanese>,
        "en": <description of action in English>
    },
    "input": {
        "type": "object",
        "properties": {
            <input arg1>: <schema of arg1>,
```

```
            <input arg2>: <schema of arg2>,
            ...
        },
        "required": <array of required args>
    },
    "schema": <schema of data>,
    "note": {
        "ja": <description of action in Japanese>,
        "en": <description of action in English>
    }
}
```

表 5-5　Action Objectの内容

Property	Type	Required	Description	Example
epc	string	No	対応するECHONET LiteのEPCをstringでHex表記する。対応するEPCが存在しない場合は省略	"0xB3"
descriptions	object	No	actionの名称	—
descriptions.ja	string	No	action名（日本語）	"一括停止設定"
descriptions.en	string	No	action名（英語）	"All stop setting"
input	object	No	actionの入力パラメータ	—
input.type	object	No	入力パラメータの型。object型固定	"object"
input.properties	object	No	入力パラメータのproperty object列	{"mode": “sleep","speed": "fast"}
input.required	array	No	必須property object列	["mode"]
schema	object	No	actionの出力を示す。出力が不要な場合は空とする	{}
note	object	No	actionに関する付加情報	—
note.ja	string	No	付加情報を日本語で記述する	"ECHONET Liteでは Set only property",
note.en	string	No	付加情報を英語で記述する	"Access rule of the corresponding ECHONET Lite property is Set only"

Event object

本版ガイドラインではEvent objectについて具体的な記述例を示さない。次版以降での追記を予定している。他の通知実現方法（ロングポーリングやWebhookを使用）については、5.10にて例示する。

5.8 機器オブジェクトのプロパティ値操作（SET/GETなど）

エコーネットでは、プロパティに対する操作をESV（ECHONET Liteサービス）として規定している。エコーネットにおけるプロパティ値をサーバ上へマッピングしてリソースとして公開し、ESVをHTTPメソッド（CRUD操作）に対応付けて操作することで、RestfulなAPI操作が可能となる。機器オブジェクトのプロパティレベルの具体的なマッピングおよび操作については、次章にて議論する。

5.9 クライアントのキャッシュの扱い

クライアントからサーバを経由して宅内機器を操作する（状態取得や制御を実施する）場合、処理によっては、応答まで長い時間を要するケースがある。HTTPキャッシュ（RFC 9111）は必須ではないが、クライアントからサーバに対して値を取得するケースにおいて、キャッシュしたリソースの再利用は有効となるため、参考として例示する。

メーカコードなどの静的データ（基本的に長期不変・固定情報）や、前回アクセスから一定期間データ更新が行われないデータ（動的情報）を扱う場合などにおけるHTTPキャッシュの活用例について以下に示す。

HTTPキャッシュのモデルには、Validation Model（検証モデル）とExpiration Model（期限切れモデル）がある。

Validation Model

データ更新されていた場合のみ取得するモデルとなる。

- サーバ：最終更新日付かエンティティタグを必要に応じて使用する

```
Last-Modified: True, 02 Jan 2018 00:00:00 GMT
ETag: "fa39b31e285573ee37af0d492aca581"
```

- クライアント：最終更新日付にて条件付きリクエストを用いる時

```
GET /elapi/v1/devices/12345 HTTP/1.1
If-Modified-Since: True, 02 Jan 2018 00:00:00 GMT
```

- クライアント：エンティティタグにて条件付きリクエストを用いる時

```
GET /elapi/v1/devices/12345 HTTP/1.1
If-None-Match: "fa39b31e285573ee37af0d492aca581"
```

- サーバ：変更なければステータスコード304（と空ボディ）、あれば200（と変更内容）を返信

Expiration Model

キャッシュの有効期限切れ時のみ取得するモデルとなる。

- サーバ：期限切れ日時か現在時刻からの秒数を指定する。

```
Expires: Wed, 03 Jan 2018 00:00:00 GMT
```

```
Cache-Control: max-age=3600
```

前者は、指定された期限までは応答で返されたデータを、クライアントがキャッシュして利用可能であることを示す。 後者は、リアルタイム性が低いケースであり、Dateヘッダからの経過秒数を指定する。 両方使う場合は、Cache-Control優先となる。 HTTPヘッダ内の日時形式（RFC 9112 (HTTP1.1)のHTTP-data記載の3種）に対応すること。

- クライアント：上記レスポンスヘッダの値を参考に、次回リクエストの発行タイミングを考える。

上記はキャッシュの有効活用を想定したが、逆に、キャッシュを全く使用しない（させない）ケースも存在する。

（明示的に）キャッシュさせたくないケース

Cache-Controlヘッダを用いることにより、クライアントに毎回アクセスを要求する（クライアントにキャッシュや再利用しないように指示する）ことができる。

- サーバ：検証が必要（現在でも有効か否かをサーバに問い合わせ、確認がとれない限り再利用してはならない）と明示

```
  Cache-Control: no-cache
```

- サーバ：キャッシュをしてはならないことを明示

```
  Cache-Control: no-store
```

5.10 機器オブジェクトのプロパティ値通知（INF）

本ガイドラインで想定するWeb APIでは、基本的にクライアントからの要求に即座に返答する通常の同期的なHTTP RESTを想定しているため、機器側のタイミングで非同期的に情報伝達を行うINFを利用するには工夫が必要である。

INFの取扱いについて

最も単純には、サーバ側において機器からINFを受信した際にその値を用いてキャッシュを更新しておき、クライアントからはそれに定期的にアクセスする（ポーリングを行う）ことでその更新を検知する方法が考えられる。このやりとりには5.9にて説明したValidation Modelを用いることができる。この手法は非常に簡便だがいくつかの問題がある。一つには情報伝達タイミングがクライアントからのリクエストのタイミングに限られるために、サーバが非同期的にINFを受け取った瞬間にその値をクライアントに伝達することができず常に遅延が生じるという点である。もう一つには、クライアントがその値に関心があるかどうかにかかわらず、サーバではINFを受け取りうるプロパティ全てに対して履歴を保存する必要があるという点である。

従って、INFを適切に処理するにはサーバ側からクライアントへメッセージを送信するプッシュ通知型の通信方式についてもサポートされることが望ましい。サーバからプッシュ通知を行う手法はいくつか知られている。

- W3C関連：
  - Push API（Web Push。単一ブラウザ通知。サーバからPush可能。Android対応、iOS非対応）
  - Service Workers
  - Web Notification（単一ブラウザ通知。Android対応、iOS非対応）
- スマホアプリ関連（ネイティブPush）：
  - APNS（Apple Push Notification Service）
  - GCM（Google Cloud Messaging）、FCM（Firebase Cloud Messaging）
- その他：
  - WebSocket
  - MQTT
  - Webhook

また、RESTで提供される手段を用いながらもコネクションを比較的長時間持続させるロングポーリングと呼ばれる方法もある。 本節では、ロングポーリングとWebSocket、MQTT、そしてWebhookによる実現事例について紹介する。

ロングポーリングによるINF実現例

本来HTTP RESTでのアクセスは、リクエストを受けたら即座にレスポンスを返すことが期待されている。「ロングポーリング」とはこのレスポンスを遅らせ、その間サーバとクライアントの間にコネクションを維持するという手法である。これにより、サーバ側で何か情報が発生した段階で即座にそれを送信することができる。通常のHTTPアクセス手段を流用して実装できる汎用性の高い方法であるが、サーバとの同時接続数が増える傾向にあるため、その対策を講じることや、切断時の再接続をうまくハンドリングする必要性が生じる。

第6章で詳細説明されるGETメソッドによるプロパティ値の取得方法を踏まえ、INF の受信タイミングをロングポーリングで受け取ることを考える。このリクエストを通常のGETと区別するため、INF

向けには他のメソッド、例えばPOSTで取得することで、サーバ側にロングポーリングをリクエストすることとする。サーバ側は、値が更新されるまではレスポンスを返さないように実装する。

■ リクエスト

```
POST /elapi/v1/devices/<device id>/properties/<Property resource name>
HTTP/1.1
```

■ レスポンス（更新されるまで返答しない。内容はGETの場合と同一である）

```
{
     <property resource name>:<data>
}
```

or

■ レスポンスがobjectの場合

```
{
    <property resource name>: {
        <element name>: <data> ,
         <element name>: <data> ,
         ...
    }
}
```

さらに、なんらかの要因により接続が切れて再接続するまでの間にサーバが機器からINFを受け取ってしまった時にも、その情報を再接続時に受け取ることを可能にすることが望ましい。これにはリクエストヘッダにIf-Modified-Sinceヘッダを付与し、その時点以降に変更が生じているならばサーバは即座にレスポンスを返すように実装すればよい。この部分はサーバ側に値を記録しておかねばならないという点で冒頭に述べたキャッシュによる実装に似通っているが、① ロングポーリングでのINF取得を仮定する場合は切断から再接続までの時間はさほど長くないと想定できるため、履歴を保存しておくべき時間を長くする必要性は低いこと、 ② POSTでアクセスされていないプロパティの値を保存する必要はないなどの点から、依然単純なキャッシュ実装より優れている。

WebSocketによるINF実現例

WebSocketをはじめコネクションベースの方法を用いる場合は、通信モデルとしてPub/Subを用いると簡便である。Pub/Subではトピックと呼ばれる文字列を介して通信を行う。このトピックとしてこれまで説明してきたリソース名を用いることで、本Web APIの基本概念を大幅に変更することなくINF向け拡張を実装することができる。

Pub/Subモデルをプロトコルレベルでサポートしている場合（MQTTやWAMPなど）では、メッセージの配送だけを専門に担当するブローカーやルーターと呼ばれるサーバを用いることが多いが、本節

で説明する、WebSocket上で実装するPub/SubではRESTからの連続性に留意し、サーバがブローカー機能を兼ね、通信はサーバとクライアント間で起こるものとする。

リクエストとレスポンスの例（サブプロトコルとして「echonet」という文字列を採用している）

■ リクエスト

```
GET /websocket  HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: echonet
Sec-WebSocket-Version: 13
```

■ レスポンス

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: echonet
```

■（クライアント）サブスクリプション

```
{
    "method": "subscribe",
    "path": "/elapi/v1/devices/<device id>/properties/operationStatus"
}
```

■（サーバ）サブスクリプションAck

```
{
    "method": "subscribeAck",
    "path": "/elapi/v1/devices/<device id>/properties/operationStatus"
}
```

■（サーバ）パブリケーション

```
{
    "method": "publish",
    "path": "/elapi/v1/devices/<device id>/properties/operationStatus",
    "value": false
}
```

■（クライアント）アン・サブスクリプション

```
{
    "method": "unsubscribe",
    "path": "/elapi/v1/devices/<device id>/properties/operationStatus"
}
```

■（サーバ）アン・サブスクリプションAck

```
{
    "method": "unsubscribeAck",
    "path": "/elapi/v1/devices/<device id>/properties/operationStatus"
}
```

MQTTによるINF実現例

MQTTはTCP上でPub/Subモデルを実装した軽量プロトコルで、IoTに適した様々な特性を持っている。その詳細は省くが、これまで説明してきた方法との重要な差異としては、MQTTネットワークの構成には、メッセージの生成と消費を行うクライアント以外にメッセージの配送を担当するブローカーが必要であるという点である。従って、本節で説明するINFの受け取りにMQTTを用いる場合にも、別にブローカーが必要となる。これまで説明してきたサーバも、そのAPIを利用するクライアントも、MQTTブローカーの観点からはMQTTクライアントであるが、サーバの方はPublishのみを行うMQTTクライアント(Publisher)、クライアントの方はSubscribeのみを行うMQTTクライアント(Subscriber)となる。

なお、MQTTはTCP上に直接実装されたバイナリのプロトコルであり、HTTPを用いていないため、正確にはWeb APIではない。Webクライアント、例えばブラウザからMQTTを利用したい場合は、AWS IoTやMosquittoなど、MQTT over WebSocketを行うことが可能なブローカーを用いる必要がある。

コンセプトは非常にシンプルで、前節で説明したWebSocketでも用いたPub/Subが素直に実装されるが、そのシーケンスはQoSと呼ばれる、メッセージの到達保証の度合いをコントロールするパラメータ、およびそれに伴う再送の回数により異なってくる。WebSocketによるINF実現例で提示したリクエストとレスポンスの例と最も近いのはQoS=1のときであるので、同様のことをMQTTで実現した場合のシーケンス図を次に示す（バイナリプロトコルであるMQTTでは、リクエストラインのようにメッセージを平易に記述する手法があるわけではない）。

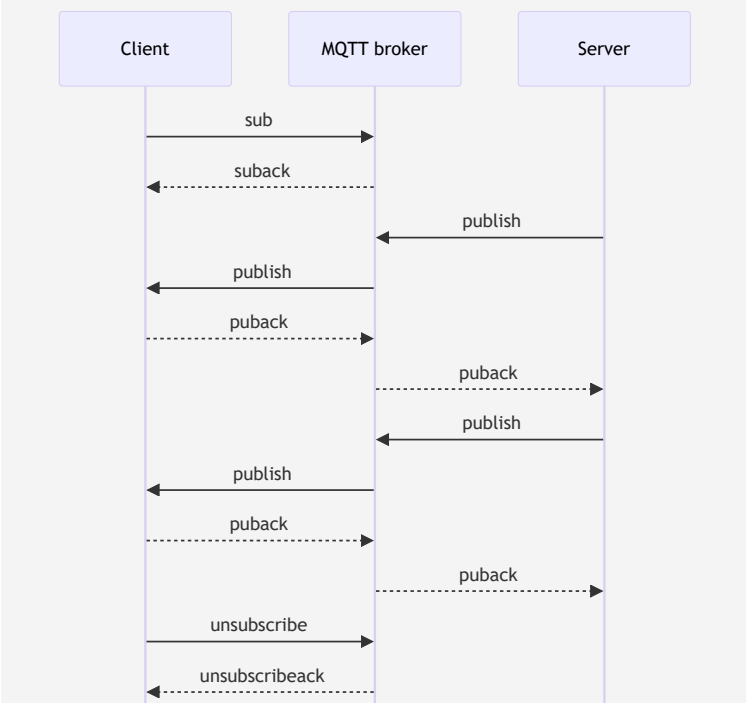


図 5-1　MQTTによるINF実現例

このように、ブローカーが挟まっている以外はほとんどWebSocketと同様である。
Publish/Subscribeを行うトピックも、WebSocketのリクエストとレスポンスの例で提示したpathの部分（RESTで用いたリソース文字列そのもの）とし、ペイロードとして適宜valueを入れて実装すればよい。

WebhookによるINF実現例

Webhookとは、クライアントがあらかじめ指定したURLにサーバがPOSTメソッドを利用して通知を行う仕組みである。

まず、サーバがwebhookをサポートしているかを確認する。レスポンスはサーバがサポートしている通知手段の情報である。以下の例は、サーバがWebhookをサポートしているが、通知の設定は初期状態の場合である。

リクエスト：サーバがWebhookをサポートしているかを確認する

```
GET /elapi/v1/notifications HTTP/1.1
```

レスポンス：Webhookをサポートしているが、subscriptionは未設定

```
{
  "webhook": {
    "subscriptions": []
  }
}
```

通知を設定する場合は、methodでsubscribeを指定し、pathで通知対象プロパティを、callBackUrlで通知先URLを指定する。通知先の認証等が必要な場合は、その情報も記述する。以下の例は、通知先のapiKeyを設定する場合である。別のリクエストを発行し、path毎に異なるcallBackUrlやapiKeyなどを設定することも可能である。pathで指定したプロパティが既にsubscribeされている場合は、callBackUrlやapikeyを上書きする。

リクエスト（subscibeするプロパティを設定）

```
POST /elapi/v1/notifications HTTP/1.1

{
  "webhook": {
    "method": "subscribe",
    "path":
"https://www.example.com/elapi/v1/devices/0123/properties/operationStatus"
,
    "callBackUrl": "https://sh-center.org/postreceive",
    "apiKey": {"key": "X-Webhook-key", "value": "0123ABC"}
  }
}
```

プロパティを指定して通知を解除する場合はmethodでunsubscribeを指定する。

リクエスト（subscribeを解除）

```
POST /elapi/v1/notifications HTTP/1.1

{
  "webhook": {
    "method": "unsubscribe",
    "path":
"https://www.example.com/elapi/v1/devices/0123/properties/operationStatus"
  }
}
```

Webhookの通知設定がされている場合に通知情報を取得する例を以下に示す。プロパティ `operationStatus`と `operationMode`はhttps://sh-center.org/postreceiveに通知し、プロパティ `faultStatus`はhttps://sh-center.org/faultStatusに通知する設定であることがわかる。

リクエスト（通知を依頼したプロパティの確認）

```
GET /elapi/v1/notifications HTTP/1.1
```

レスポンス

```
{
  "webhook": {
    "subscriptions": [
      {
        "path":
"https://www.example.com/elapi/v1/devices/0123/properties/operationStatus"
,
        "callBackUrl": "https://sh-center.org/postreceive",
        "apiKey": {"key": "X-Webhook-key", "value": "0123ABC"}
      },
      {
        "path":
"https://www.example.com/elapi/v1/devices/0123/properties/operationMode",
        "callBackUrl":"https://sh-center.org/postreceive",
        "apiKey": {"key": "X-Webhook-key", "value": "0123ABC"}
      },
      {
        "path":
"https://www.example.com/elapi/v1/devices/0123/properties/faultStatus",
        "callBackUrl": "https://sh-center.org/faultStatus",
        "apiKey": {"key": "X-Webhook-key", "value": "456XYZ"}
      }
    ]
  }
}
```

通知されるデータはこのようになる

```
POST https://sh-center.org/postreceive HTTP/1.1

X-Webhook-key: 0123ABC

{
  "path": "/elapi/v1/devices/0123/properties/operationStatus",
  "body": {"operationStatus": true}
}
```

複数のプロパティを一括してsubscribeもしくはunsubscribeするには、第7章記載のbulksを利用する。

次に、もう1つ実現例を説明する。 通知先のURLの登録、解除は何らかの手段（例えば上記の実現例の手順）で行われていることを前提に、WebhookによるINFの通知方法を説明する。

HTTPメソッドは、例えばPOSTメソッドを使用する。また、リクエストがWebhookであることと対象のリソース名を通知する。リソース名の通知には、例えば、リクエストのヘッダフィールドやボディにリソース名を記述する方法がある。ヘッダフィールドにリソース名をセットする場合、ヘッダ名は、例えばX-Elapi-notificationとする。ヘッダフィールドやボディに記述するリソース名は、対象プロパティ・URLの登録時にクライアントから指定されたリソース名を使用するなどして、クライアントが識別可能なものとする。以下では、ボディにリソース名を記述する例として、resourceキーに対して、resource URLを記述するものとしたが、device ID、propertyなどのキーを設けるものとしてもよい。

クライアントが複数のサーバからINFを受信する場合、複数のサーバのリソース名（device ID）が重複するケースが考えられる。そのため、①クライアントが、対象プロパティ・URLの登録時に、サーバごとに異なるURLを指定する、②INFのHTTPリクエストのヘッダあるいはボディにサーバを識別する識別子を記述する、などの対応によりクライアントがリソースを一意に特定することができるようにする。サーバを識別する識別子としては、例えば、ホスト名や、対象プロパティ・URLの登録の際のHOSTヘッダの内容を用いる。以下の例では、クライアントがサーバごとに異なるURLを指定しているものとして、INFのHTTPリクエストのヘッダ、ボディにはサーバを識別する識別子を記載しないものとする。

WebhookによるINFの例（ヘッダフィールドにリソース名を記述する場合）

■ リクエスト

```
POST <クライアントが登録した任意のURL> HTTP/1.1

X-Elapi-notification: /elapi/v1/devices/<device_id>/properties/<Property
resource name>
Content-Type: application/json

{
  <property resource name>: <data>
}
```

■ レスポンス

```
リクエストと同様
```

WebhookによるINFの例（ボディにリソース名を記述する場合）

■ リクエスト

```
POST <クライアントが登録した任意のURL> HTTP/1.1

Content-Type: application/json
```

```
{
  "events": [
    {
      "resource": <resource URL>,
      "value": <data>
    }
  ]
}
```

■ レスポンス

```
リクエストと同様
```

一般照明の例（ヘッダフィールドにリソース名を記述する場合）

■ リクエスト

```
POST <クライアントが登録した任意のURL> HTTP/1.1

X-Elapi-notification:
/elapi/v1/devices/<device_id>/properties/operationStatus
Content-Type: application/json

{
  "operationStatus": true
}
```

■ レスポンス

```
リクエストと同様
```

一般照明の例（ボディにリソース名を記述する場合）

■ リクエスト

```
POST <クライアントが登録した任意のURL> HTTP/1.1

Content-Type: application/json

{
  "events": [
    {
```

```
      "resource":
"/elapi/v1/devices/<device_id>/properties/operationStatus",
      "value": true
    }
  ]
}
```

■ レスポンス

リクエストと同様

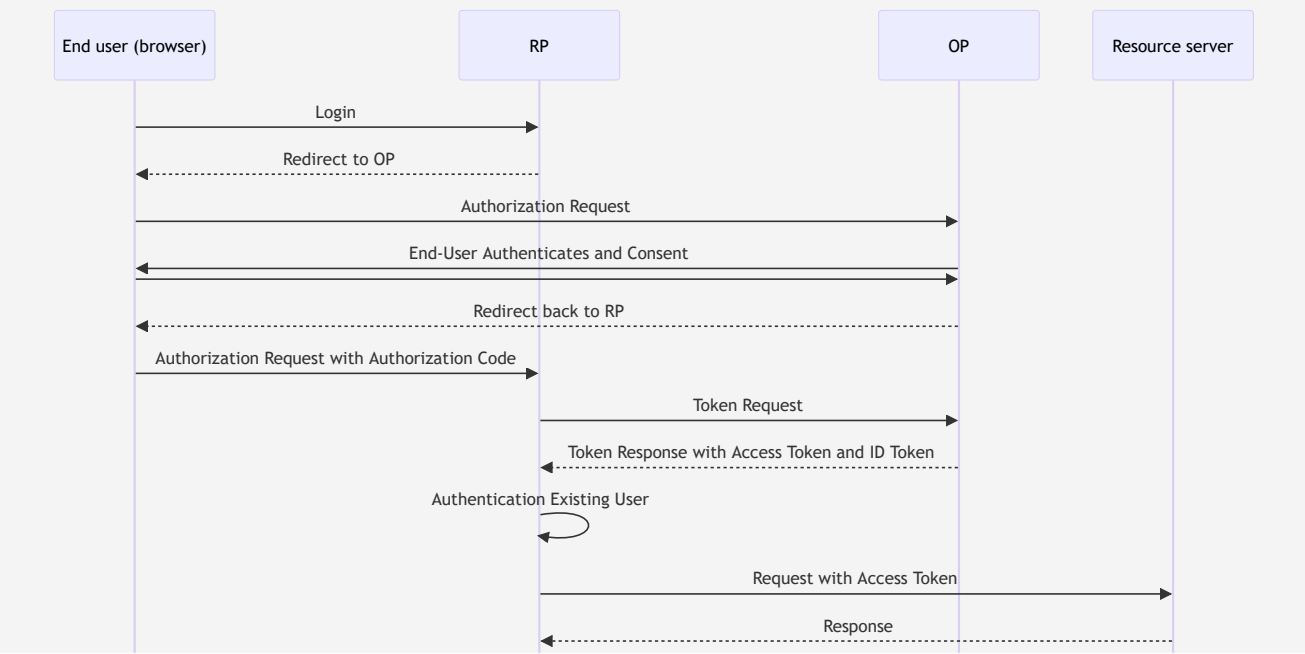
5.11 認証/認可（事例紹介）

API利用者の特定や、呼び出し回数のカウントにより過剰なアクセス時に通信遮断するなど、認証が必要となるケースは多いと考えられる。 本Web APIガイドラインでは、認証・認可の仕組みについて参考事例として紹介する。Web APIにおける認証・認可方式は、トークン認証やOAuth2.0認可などが挙げられるが、どの方式を採用するかは、ユースケースによって異なりうる。

ここでは、OpenID ConnectやOAuth2.0を使用し、典型的な事例として、本Web APIをスマートホンなどから使用する場合（Authorization Code）と、サーバ間で使用する場合（Client Credentials）の場合の2種類について例示する。

Authorization Codeによる実現例

まず、Authorization Codeの事例について示す。 図中、RPはRelying Party、OPはOpenID Provider（もしくはIdentity Provider（IdP)とも呼ぶ）。リソースサーバは本Web APIを実装したサーバとなる。



エンドユーザ（ブラウザ）はRPへログインを試みると、RPからOPへのリダイレクトメッセージが返却される。

RPからのレスポンス例：

```
HTTP/1.1 302 Found
Location: https://server.example.com/authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcallback
```

state以外は必須。scopeにはopenidを含まねばならず、Authorization Code Flowを使用する場合はresponse_typeは codeとなる。 続いて、エンドユーザ（ブラウザ）は、下記のようにOPへ認可リクエストを送信する。

エンドユーザからの認可リクエスト例：

```
GET /authorize?
     response_type=code
     &scope=openid%20profile%20email
      &client_id=s6BhdRkqt3
      &state=af0ifjsldkj
      &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcallback HTTP/1.1

Host: server.example.com
```

OPはこの認可リクエストを検証し、有効な場合、エンドユーザの認証を試みる。通常、認証用のユーザインタフェースを表示し、ログイン（ユーザ名、パスワードなど）による認証など実施し、認証後にRPへ情報を送ることに対してエンドユーザの同意を得る。 同意獲得後、OPはエンドユーザ（ブラウザ）へリダイレクトを含むレスポンスを返却する。

OPからのレスポンス例：

```
HTTP/1.1 302 Found
Location: https://client.example.org/callback?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

エンドユーザ（ブラウザ）からリダイレクトにて呼び出されたRPは、OPへトークン要求を発行する。

RPからのリクエスト例：

```
POST /token HTTP/1.1

Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcallback
```

OPは、認可コード（code）の内容を検証し、有効なクレデンシャルであれば下記のようにアクセストークンやIDトークンなどを返却する。

OPからのレスポンス例：

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciO……"
}
```

RPは、ヘッダにアクセストークンを伴い、リソースサーバのWeb APIを呼び出す。

RPのリクエスト例：

```
GET /elapi/v1/devices HTTP/1.1

Host: api.xxx.com
Authorization: Bearer  SlAV32hkKG
```

Client Credentialsによる実現例

次に、Client Credentialsの事例について示す。 この場合、クライアントは通常、デーモンサービスやWebサイトなどが対象となる。

認可サーバのエンドポイントとして、例えば「`/oauth2/token`」をTLSで保護されたPOSTメソッドにて呼び出す。リクエストヘッダには、client_idとclient_secretの値をbase64でエンコーディングした属性を含む（HTTP Basic 認証：RFC 7617）。

クライアントのリクエスト例：

```
POST /oauth2/token HTTP/1.1

Host: www.auth-server.com
Accept: application/json
Cache-Control: no-cache
Authorization: Basic eW91clfaWQ6eW91cl9jbG……
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=xxxx
```

eW91clfaWQ6eW91cl9jbG……（後半省略） は、base64(<client_id>:<client_secret>)の値。リクエストボディにて、grant_typeに client_credentialsを指定する。スコープ xxxxはアクセストークン取得に必要なアクセスレベルであり、省略可能。

認可サーバは、client_idとclient_secretを検証し、有効なクレデンシャルであればアクセストークン（eyJraWQiOiJ……：後半省略）を返却する。

認可サーバのレスポンス例：

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token": "eyJraWQiOiJ……",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

クライアントは、ヘッダにアクセストークンを伴い、リソースサーバのWeb APIを呼び出す。

クライアントのリクエスト例：

```
GET /elapi/v1/devices HTTP/1.1

Host: api.xxx.com
Authorization: Bearer eyJraWQiOiJ……
```

リソースサーバは、アクセストークンの検証が成功すると、要求されたデータを返却する。アクセストークンの検証は、認可サーバのintrospectionエンドポイントに問い合わせるか、事前に認可サーバよりキャッシュしておいた公開鍵を使うなどして行うことができる。

第6章 ECHONET Lite仕様のマッピング指針

本章ではECHONET Liteのフレームや機器オブジェクトの各仕様をどのようにマッピングすべきか、その指針について示す。

6.1 ECHONET Liteフレームのマッピング

ECHONET Liteは、下位通信層非依存の通信プロトコルである。一般的に、IPv4やIPv6使用時にはトランスポート層にUDP（必須）、TCP（オプション）を用いて、ECHONET Lite フレーム（表 6-1）をペイロードに搭載して通信を行う。

表 6-1 ECHONET Lite フレームの対応

フィールド	説明	Web APIでの対応
EHD	ECHONET Liteヘッダ	非サポート
TID	Transaction ID	非サポート
SEOJ	Source ECHONETオブジェクト	非サポート
DEOJ	Destination ECHONETオブジェクト	機器オブジェクト（インスタンス）のidにて識別
ESV	ECHONET Liteサービス	後述
OPC	処理対象プロパティカウンタ	非サポート
EPC	ECHONETプロパティ	後述
PDC	プロパティデータカウンタ	非サポート
EDT	ECHONETプロパティ値データ	後述

表 6-1に、ECHONET Liteフレームを構成する各フィールドに関する本Web APIでの対応指針について記載する。主に、DEOJ（相手先機器オブジェクト）とESV（サービス種）、EPC（プロパティ）、EDT（プロパティ値データ）を対象にマッピングすることで、Web APIへの変換を実現する。EHDやTID、PDCは宅内のコントローラと機器間でのみ関連する事項とし、サーバから外部へは公開されない情報とする。TIDは、コントローラと機器間の通信管理に使用され、クライアントとサーバ間のセッション管理の一部として扱うことも考えられる。しかしながら、クライアントからのリクエストに対して、サーバ側がコントローラや機器に対して再送を含む様々な形式で通信を実施するケースも考えられ、サーバ実装に依存することから、TIDは本Web APIへのマッピングのスコープ外とする。

6.2 DEOJのマッピング

ECHONET Liteのオブジェクトは、ノードプロファイルオブジェクトと機器オブジェクトの二種類が存在するが、機器オブジェクトのみ扱う。これは、Web APIを介して操作を行うクライアント側アプリケーションは通常、機器オブジェクトへの操作が主体となるためである。通信部の確認などノードプロファイルオブジェクトへの操作が必要になるケースも考えられるが、本バージョンのガイドラインではスコープ外とする。 コントローラは機器の一種として扱われる。

下記では、複数の機器を指定する場合の記述例について示し、一覧取得に使用するものとする。なお、一括制御に適用する複数機器の指定方法については、第7章のbulksやgroupsを参照すること。

コントローラすべてを指定したい場合：

全コントローラを指定したい場合は、後述の特定機種すべてを指定したい場合の方法に従いクエリーパラメータとして type=controllerを指定する、もしくは、サーバが controllerというサービス種をサポートする場合は「/elapi/v1/<サービス指定（省略可）>/controllers」と指定することができる。

機器すべてを指定したい場合：

全機器（全コントローラを含む）を指定したい場合は、「/elapi/v1/<サービス指定（省略可）>/devices」と指定することができる。

特定機種すべてを指定したい場合：

ある機種をすべて指定したい場合は、「/elapi/v1/<サービス指定（省略可）>/devices?type=homeAirConditioner」のように、機種のタイプを指定すればよい。

インスタンスコードが0x00の場合の扱い：

ECHONET Liteでは、あるノード上のある機種オブジェクトに関して、インスタンスコードを0x00とすることで、同機器オブジェクトのインスタンスすべてを指定可能であるが、本Web APIへのマッピングは行わないものとする。

6.3 ESVのマッピング

ECHONET LiteにおけるESVは、表 6-2 のように定義されている。

表 6-2　ESVの対応

ESV	サービス内容	記号	Web APIでの対応
0x60	プロパティ値書き込み要求（応答不要）	SetI	PUTメソッド（リクエスト）
0x61	プロパティ値書き込み要求（応答要）	SetC	
0x62	プロパティ値読み出し要求	Get	GETメソッド（リクエスト）
0x63	プロパティ値通知要求	INF_REQ	後述
0x6E	プロパティ値書き込み・読み出し要求	SetGet	PUTメソッド（リクエスト）
0x71	プロパティ値書き込み応答	Set_Res	PUTメソッド（レスポンス）
0x72	プロパティ値読み出し応答	Get_Res	GETメソッド（レスポンス）
0x73	プロパティ値通知	INF	後述
0x74	プロパティ値通知（応答要）	INFC	後述
0x7A	プロパティ値通知応答	INFC_Res	後述
0x7E	プロパティ値書き込み・読み出し応答	SetGet_Res	PUTメソッド（レスポンス）

ESV	サービス内容	記号	Web APIでの対応
0x50	プロパティ値書き込み要求不可応答	SetI_SNA	PUTメソッド（レスポンス）
0x51	プロパティ値書き込み要求不可応答	SetC_SNA	PUTメソッド（レスポンス）
0x52	プロパティ値読み出し不可応答	Get_SNA	GETメソッド（レスポンス）
0x53	プロパティ値通知不可応答	INF_SNA	後述
0x5E	プロパティ値書き込み・読み出し不可応答	SetGet_SNA	PUTメソッド（レスポンス）

ESVの種類は、要求、応答、不可応答に大別されるが、Set操作やGet操作をHTTPメソッドへマッピングする場合、要求（リクエスト）と応答・不可応答（レスポンス成功・失敗）に対応付けることができる。

Set系ESVは、値の更新や制御に相当するため、PUTメソッドへ対応づける。SetI, SetC, SetGetの3種類は区別なく、応答にてプロパティ値も返却される形式に統一し、PUTメソッドによるリクエストとレスポンスを使用する。また、AIF認証仕様にて規定されている機器の応答待ち時間に関連し、Ackの返却、タイムアウト処理などをどのようにマッピングするかは、本バージョンのガイドラインではスコープ外とする。

Get系ESVは、値の取得・読み出しであるため、GETメソッドへ対応づける。サーバ上にキャッシングされるデータの扱いについては後述する。 INF系ESVは、通常のWeb APIではプル・ベースとなるため、リアルタイム通知を受けることは難しいが、擬似的にGETメソッドを用いてポーリングすることや、WebSocket、MQTT、Web Notification API、Webhookなどのプッシュ・ベース通信モデルを活用することで実現する。

6.4 EPC、EDTのマッピング

EPC、EDTは、機器オブジェクト仕様とリンクして定義づけられている。EPCは、URIにてリソース定義され、EDTは、リクエストボディやレスポンスボディにてやりとりされる。具体的には、5.7記載のDevice Descriptionをベースに、GET可能なプロパティは、「/elapi/v1/devices/<device id>/propertie/<property resource name>」へマッピングされる。プロパティ名称はプロパティリソース名に変換され、「機器仕様部」（別書）にて詳細規定されている。SET対象となるプロパティは、GET可能な場合は properties配下のプロパティリソース名にて扱われるが、GETできない場合は、actions配下に操作名称を別途定義して扱うことになる。

次節以降、詳細に見ていく。

なお、以降、エンドポイントを示すURIのうち、prefix部（/elapi/v1）は表記を簡潔にする都合、省略する場合がある。

6.5 ECHONET Lite機器オブジェクトのマッピング方式および操作

機器オブジェクトの操作に関する基本APIを表 6-3に示す。

表 6-3　機器オブジェクトの操作に関するAPI

http method	path	description

http method	path	description
GET	/devices/<device id>/properties	GET対象プロパティ値の一括取得
GET	/devices/<device id>/properties/<property resource name>?<query>	指定プロパティの値取得
PUT	/devices/<device id>/properties/<property resource name>	指定プロパティの値設定
PATCH	/devices/<device id>/properties	SET対象プロパティ値の複数設定
POST	/devices/<device id>/echoCommands	指定プロパティ値の設定

ECHONET Liteでは、二種類のオブジェクト（ノードプロファイルオブジェクトと機器オブジェクト）を定義しているが、本Web APIガイドラインでは、前述の通り、機器オブジェクトのみを扱う。機器オブジェクトのスーパークラスは、基本的に個々のプロパティがマッピングされるが、プロパティマップなど、JSON化により自明となる情報は省略される。スーパークラスのプロパティのマッピングについては、「機器仕様部」（別書）参照のこと。

以下、機器オブジェクトの各種操作APIについて基本方針を示す。

GET /devices /<device id >/properties

「/elapi/v1/<サービス指定（省略可）>/devices/<device id>/properties」を指定しGETすることで、GET対象となるすべてのプロパティについて値を取得可能とする。返却されるデータは事前にサーバにてキャッシュされた値であっても良い。

■ リクエスト

```
GET /elapi/v1/devices/<device id>/properties HTTP/1.1
```

■ レスポンス

```
{
  <property resource name>: <property value>,
  <property resource name>: <property value>
    ...
}
```

一般照明の例

■ リクエスト

```
GET /elapi/v1/devices/<device id>/properties HTTP/1.1
```

■ レスポンス

```
{
    "operationStatus": true,
    "faultStatus": false,
    "brightness": 50,
    "operationMode": "color",
    "rgb": {"r": 20, "g": 255, "b": 0 }
}
```

properties内の複数プロパティリソース指定方法

> propertiesは機器に関するすべてのプロパティリソースを対象とするが、サーバ上のキャッシュを用いず宅内機器の全プロパティへGETによる値取得操作を実施する場合、時間を要する処理となる可能性がある。クライアント側で対象機器のすべてのプロパティリソースではなく、いくつかのプロパティリソースの値を取得したい場合、後述の個別プロパティリソース指定によるAPIを用いて都度取得することも可能だが、クエリーパラメータを用いて所望のプロパティを複数指定できる。 具体的には、下記のように propertyNamesというキーに対して、所望のプロパティリソース名を必要に応じて複数、コンマ区切りで指定すれば良い。なお、RFC 3986 のセクション 3.4 Queryでは複数の値を持つパラメーターの定義はなく、一般的には実装依存とされている。下記以外の指定形式も存在するが、サーバにて使用するWebフレームワークや処理系が対応している必要がある。下記の例は、propertyNamesの値"propX,propY"を取得し、「,」区切りで各指定プロパティリソースを抽出することで比較的容易に実装できる。

複数プロパティリソース指定の例

■ リクエスト

```
GET /elapi/v1/devices/<device id>/properties?propertyNames=propX,propY
HTTP/1.1
```

■ レスポンス

```
{
    "propX": value1,
    "propY": value2
}
```

GET /devices/<device idgt;/properties/<property resource name>?<query>

「/elapi/v1/<サービス指定（省略可）>/devices/<devic id>/properties/<property resource name> ? <query>」を指定しGETすることで、指定したプロパティ値を取得可能とする。queryは通常は不要だが、スマートメータなど一部機器の特定プロパティでSET(EL)とGET(EL)のatomic operationが必要な場合にSETのdataを指定するためにqueryを利用する。keyはproperty毎に定義する。

■ リクエスト

```
GET /elapi/v1/devices/<device id>/properties/<property resource name>?
<query> HTTP/1.1
```

■ レスポンス

```
{
     <property resource name>: <data>
}
```

or

■ レスポンスがobjectの場合

```
{
    <property  r esource name>: {
        <element name>: <data>,
        <element name>: <data>,
        ...
    }
}
```

一般照明の例

■ リクエスト

```
 /elapi/v1/devices/<device id>/properties/operationMode HTTP/1.1
```

■ レスポンス

```
{
    "operationMode": "color"
```

```
}
```

■ リクエスト

```
GET /elapi/v1/devices/<device id>/properties/rgb HTTP/1.1
```

■ レスポンスがobjectの場合

```
{
    "rgb": {
        "r": 20,
        "g": 255,
        "b": 0
    }
}
```

■ リクエストにqueryがある場合

```
GET /elapi/v1/devices/<device
id>/properties/normalDirectionIntegralElectricEnergyLog1?day=0 HTTP/1.1
```

■ レスポンス

```
{
    "normalDirectionIntegralElectricEnergyLog1": {
        "day": 0,
        "energy": [
            20,
            34,
            59,
            109
        ]
    }
}
```

PUT /devices/<device id >/properties/< property resource name >

「/elapi/v1/<サービス指定（省略可）>/devices/<device id>/properties/<property resource name>」を指定しPUTすることで、指定したプロパティ値を設定可能とする。サーバは

コントローラを経由して対象機器のプロパティリソースに対応するプロパティに対してECHONET LiteのSET操作後に更にGET操作した値を取得し、クライアントへ返却する。

■ リクエスト

```
PUT /elapi/v1/devices/<device id>/properties/<property resource name>
HTTP/1.1

{
     <property resource name>: <data>
}
```

or

■ リクエストがobjectの場合

```
{
    <property resource name>: {
        <element name>: <data>,
        <element name>: <data>,
        ...
    }
}
```

■ レスポンス

```
リクエストと同様
```

一般照明の例

■ リクエスト

```
PUT /elapi/v1/devices/<device id>/properties/operationMode HTTP/1.1

{
    "operationMode": "color"
}
```

■ レスポンス

```
{
    "operationMode": "color"
}
```

■ リクエストがobjectの場合

```
PUT /elapi/v1/devices<device id>/properties/rgb HTTP/1.1

{
    "rgb": {
        "r": 20,
        "g": 255,
        "b": 0
    }
}
```

■ レスポンス

```
{
    "rgb": {
        "r": 20,
        "g": 255,
        "b": 0
    }
}
```

上記手順は、「機器仕様部」（別書）記載の各機器を対象に、Device Descriptionの内容に基づいた操作を実行することを想定している。詳細については「機器仕様部」（別書）を参照すること。

一方、「機器仕様部」（別書）で規定するDevice Description内に所望のプロパティ定義が存在しない場合や、ベンダ独自のメーカ依存コードを使用する場合など、本ガイドラインの規定範囲外となるEPC操作を可能にする補完機能を導入する（オプション扱い。7.5節にて後述）。

PATCH /devices/<device id>/properties

RFC5789（PATCH Method for HTTP）にサーバが対応している場合、「/elapi/v1/<サービス指定（省略可）>/devices/<device id>/properties」を指定しPATCHすることで、指定した複数のプロパティ値を設定可能とする。指定されなかった他のプロパティは元のままである。

リクエストに設定できないプロパティが含まれることが、サーバで判定できる場合（設定するプロパティ値が範囲外であったり、プロパティリソース名が存在しないなど）は、HTTPレスポンスのステータスコードとして400番台を返す。この場合、サーバからコントローラへのリクエストの送信は行われない。

リクエストに設定できないプロパティが含まれることが、コントローラや機器の処理時に明らかになった場合（機器側の内部状態により、プロパティの更新ができないなど）は、HTTPレスポンスのステータスコードとして500番台を返す。

必要に応じてエラー応答をレスポンスボディ（JSON形式）にて返却可能とする（オプション）。設定できなかったプロパティは、PATCHで指定したプロパティ値と共に、エラータイプとメッセージ（任意の文字列）を組として応答する。

■ リクエスト

```
PATCH /elapi/v1/devices/<device id>/properties HTTP/1.1

{
    <property resource name>: <property value>,
    <property resource name>: <property value>
    ...
}
```

■ レスポンス

```
{
    <property resource name>: <propertyValue>,  // PATCH成功プロパティ
    <property resource name>: <propertyValue>,
    ...,
    "errors": [ <error object>, <error object>...  // PATCH失敗プロパティ（オプション）
    ]
}
```

■ error object

```
{
    <property resource name>: <property value>,  // PATCH時に指定されたプロパティ
    "type": <error type>,
    "message": <error message>
}
```

一般照明の例

■ リクエスト　（範囲値外の値や、無効なプロパティが含まれる場合）

```
PATCH /elapi/v1/devices/<device id>/properties HTTP/1.1

{
    "operationMode": "color",        // 正常
    "rgb": {
        "red": 20,
        "green": 300,        // 範囲値外の値
        "blue": 0
    },
}
```

■ レスポンス

```
HTTP/1.1 400 Bad Request

{
    "operationMode": "color",        // 正常値　（400番台のエラーなので、リクエスト送信はされない）
    "errors": [  // 異常値　（PATCH要求時と同じプロパティ値を返す）
        {
            "rgb": {
                "red": 20,
                "green": 300,
                "blue": 0
            },
            "type": "rangeError",
            "message": "’green’ value is out of range."
        }
    ]
}
```

■ リクエスト　（機器から一部のプロパティが処理できず不可応答が返ってきた場合）

```
PATCH /elapi/v1/devices/<device id>/properties HTTP/1.1

{
    "operationMode": "color",
    "rgb": {
        "red": 20,
        "green": 128,
         "blue": 0
    }
}
```

■ レスポンス

```
HTTP/1.1 500 Internal Server Error

{
    "operationMode": "color",  // 設定成功
    "errors": [  // 設定失敗
        {
            "rgb": {  // 応答時のプロパティ値は実装マター　（本例はPATCH要求時と同じプロパティ値）
                "red": 20,
                "green": 128,
                 "blue": 0
            },
            "type": "deviceError",
            "message": "SetC_SNA"
        }
    ]
}
```

6.6 Action

action（アクション）を呼び出すには、対象アクションリソース名を含むURIに対してPOSTメソッドを実行する。アクションは、機器の操作やサービス操作のうち、プロパティリソースを定義して行う操作として定義しづらい場面などで使用される。

POST /<device id>/actions/<action resource name>

■ リクエスト

```
POST /elapi/v1/devices/<device id>/actions/<action resource name> HTTP/1.1

{
  <input arg1>: <data1>,
  <input arg2>: <data2>,
  ...
}
```

■ レスポンス

```
HTTP/1.1 200 OK

<output data>
```

6.7 エラー処理

HTTPでは、レスポンス時のステータスコードを表 6-4のように規定している。これらを考慮し、サーバは適切にエラーコードなどを返却すべきである。

表 6-4 HTTP Status Code

Status Code	名称	意味
100番台	Informational	処理が継続中
200番台	Successful	正常終了
300番台	Redirection	転送の要求
400番台	Client Error	クライアント原因のエラー
500番台	Server Error	サーバ（宅内コントローラや機器との通信含む）原因のエラー

より詳細なエラー情報を返却できる場合には、表 6-5を参考に適切なステータスコードを返却しても良い（オプション）。

表 6-5 HTTP Status Codeの活用事例

Status Code	名称	意味（例）
200	OK	リクエストした処理が成功
201	Created	リクエストが成功しリソースが作成された
204	No Content	リクエストは正常に処理されたが、返すべきコンテンツが存在しない（DELETE時の成功など）
301	Moved Permanently	リソースは恒久的に移動
304	Not Modified	リソースは未更新
400	Bad Request	リクエストが不正、データ形式間違い
401	Unauthorized	認証が必要
404	Not Found	リソース（該当するパスやプロパティ）が存在しない。
405	Method Not Allowed	リソースに指定されたメソッドが許可されない
406	Not Acceptable	Acceptヘッダとマッチしない
409	Conflict	リソースが矛盾（IDが衝突など）
415	Unsupported Media Type	データ形式は正しいがサーバが対応しない

Status Code	名称	意味（例）
500	Internal Server Error	サーバサイドでエラーが発生
503	Service Unavailable	サーバが一時的に停止

本ガイドラインでは、上記ステータスコードが4XX系、5XX系の場合は更に、必要に応じてECHONET Lite Web APIの呼び出しや、サービスに起因するエラー応答もレスポンスボディ（JSON形式）にて返却可能とする（オプション）。

下記に示す通り、エラータイプとメッセージ（任意の文字列）を組として応答する。

```
{
    "type":<error type>,
    "message":<error message>
}
```

表 6-6　サービスレベルのエラー応答

Property	Type	Required	Description	Example
type	string	Yes	ErrorのTypeを示す。サーバがErrorと判断する場合（rangeError/ referenceError/typeError/timeoutError）と機器がエラーと判断する場合（deviceError）がある。	"rangeError"
message	string	Yes	ERRORの詳細を記述する任意のString data	

typeには、下記種類を想定する。

表 6-7　エラータイプの種類

クライアント（クライアント）に起因

ErrorType	Description	Example
rangeError	設定する値が仕様の範囲外の場合	number, integer：値がminとmaxの間にない場合 enum: 値が存在しない場合
referenceError	対象とするdevice IDやproperty resource nameが存在しない場合	
typeError	設定する値の型がDevice Descriptionに記述された型と一致しない場合	

サーバ（コントローラや機器の通信含む）に起因

ErrorType	Description	Example
timeoutError	機器から一定時間内に返答がない場合	通信系エラー
deviceError	機器から受信したデータがerrorに対応する値の場合。機器からSNAを受信した場合	SetGet_SNA, Set_SNAを受信した場合

例

```
{
    "type": "rangeError",
    "message": "data is out of range"
}
{
    "type": "rangeError",
    "message": "no value matches to the data"
}
{
    "type": "referenceError",
    "message": "device name is wrong"
}
{
    "type": "referenceError",
    "message": "property resource name is wrong"
}
{
    "type": "typeError",
    "message": "data should be boolean"
}
{
    "type": "timeoutError",
    "message": "timeout !"
}
{
    "type": "deviceError",
    "message": "undefined"
}
{
    "type": "deviceError",
    "message": "GET_SNA"
}
```

6.8 機器情報

「機器仕様部」（別紙）にて記載する。

第7章 応用サービス機能

これまで紹介してきた仕様は、主にECHONET Liteプロトコル自身が有する基本機能をWebサービスへマッピングした際の展開モデルが中心だった。本章では、こうした基本機能の組み合わせや別の情報の追加・データ加工などによって、より便利な応用サービスを実現するための事例について紹介していく。

なお、以降、リソースを表すパスのうち、接頭語となる″/elapi/v1″については記載を省略する。

7.1 複数命令の一括指示（bulks）

サーバは、任意のリソースを対象とした命令を複数列挙した命令セット（bulk）をクライアントからの要求に基づき作成し登録することで、一括で複数命令を指示・実行する機能（bulk実行）をクライアントに対して提供する。

図 7-1 を用いて、bulk実行に関する一連の動作について例示する。

- bulk実行のためには、まず、①クライアントが、対象とする命令（method, path, [body]（bodyは省略可能）から構成）を複数列挙した命令セット（bulk）をサーバへ登録申請する。サーバは、②同bulkを登録後、③クライアントへ同bulkの識別子となるbulk IDを返却する。
- 次に、④クライアントが同bulk IDを指定して executeアクションを指示することで、⑤サーバにて同bulk実行の有効期間が開始となり、⑥サーバはクライアントへexecution IDを返却する）。
- 以降、⑦サーバは、宅内・構内のコントローラ（機器含む）へbulkを構成する各命令を送信し実行結果の取得を進める（⑧⑨）。
- ⑩クライアントがexecution IDを指定してサーバに対してbulk実行のレスポンス取得を試みる（⑪にてサーバは応答する）。クライアントは、この操作を必要に応じてbulk実行が完了するまで複数回実施する。
- ⑫最後の命令実行が完了した後、⑬サーバでのbulk実行は完了する。以降、⑭クライアントがbulk実行のレスポンス取得を試みたとき、取得したレスポンス内の″processStatus″は終了を示す値となる（⑮）。ただし、サーバ規定によるbulk実行有効期間を超えてクライアントが同レスポンス取得を試みた場合は、サーバよりエラーが返される。
- このbulk実行有効期間は、サーバにより自由に規定可能とするが、通常は、bulk実行が完了し、クライアントによる getResultsアクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。
- なお、bulk IDは再利用可能なため、再度bulk実行を行いたい場合は、④以降の手順を繰り返し（新たなexecution IDを獲得し）実行することができる。
- 最終的に、登録したbulkが不要となった場合は、bulk IDを指定してサーバ登録されたbulkを消去する（⑰⑱⑲）。

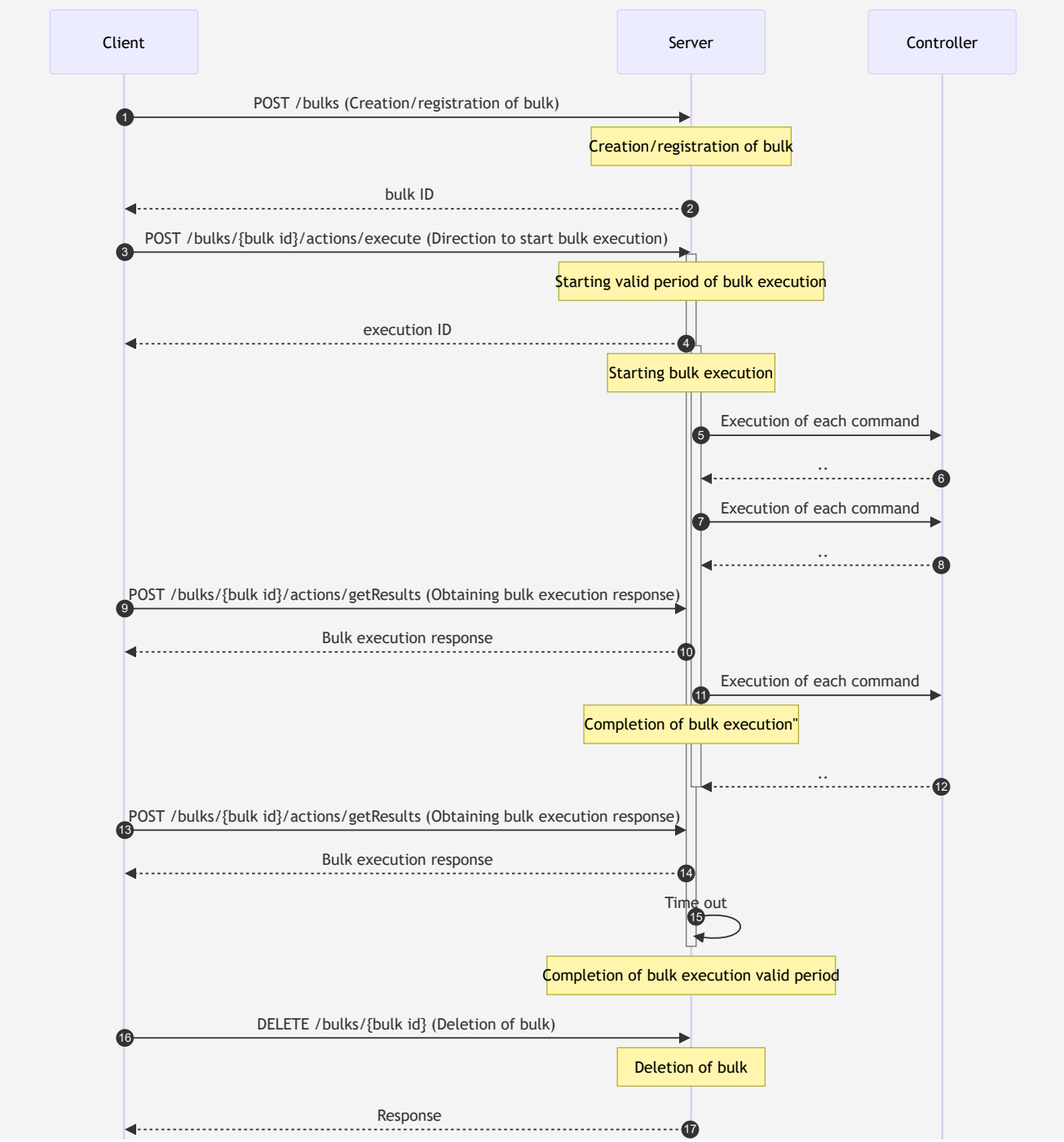


図 7-1　bulk実行の全体概要

サーバが宅内・構内のコントローラ（機器含む）へ命令実行する際、bulkの処理モード（"processMode"）は二種類のうちいずれかを指定することができる（図 7-2）。①のbulk登録時に"concurrent"モードを指定した場合は、"requests"キー内の配列にて指定した順に命令を（応答を待たずに）実行し、"sequential"モードを指定した場合は、"requests"キー内の配列にて指定した順に命令を（正常応答を待ちつつ）実行する。

"processMode"無指定の場合は、"concurrent"モードにて実行される。

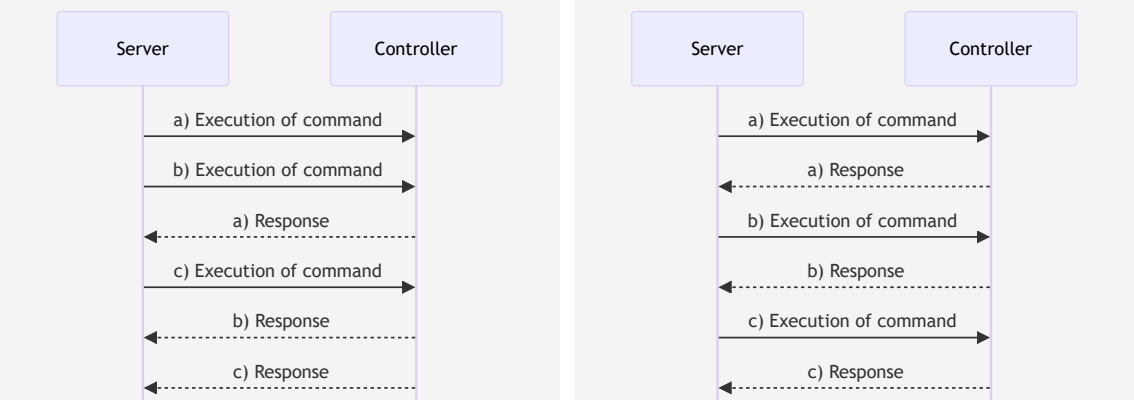


図 7-2　concurrentモード（非同期）とsequentialモード（同期）

⑪におけるbulk実行のレスポンス内容より、bulk実行状況について個別および全体の視点から確認することができる。

個別状況は、"responses"キー内の配列に列挙される各命令について（"requests"キーで指定した配列の順にて）、各命令の実行状況を"progress"キーによって確認できる。

値は、unexecuted（未実行。初期値）、executing（実行中）、completed（成功完了）、failed（失敗完了）、timeout（タイムアウト）、aborted（中断）のいずれかの値となる。

"processMode"が"concurrent"の場合は、クライアントから明示的に abortされるか、サーバ側での処理時間が（サーバ規定の）全体タイムアウト値を超えない限り、全ての命令が実行される。

図 7-3に"concurrent"モードでの処理について例示する。図では、命令a）から e）を含んだbulk内の各命令を順次実行中であり、a）は成功、b）は失敗に終わり、c）の実行でタイムアウト、d）は実行中で、e）は未実行の状態であることを示している（クライアントが呼び出した時点はd）実行中の位置）。

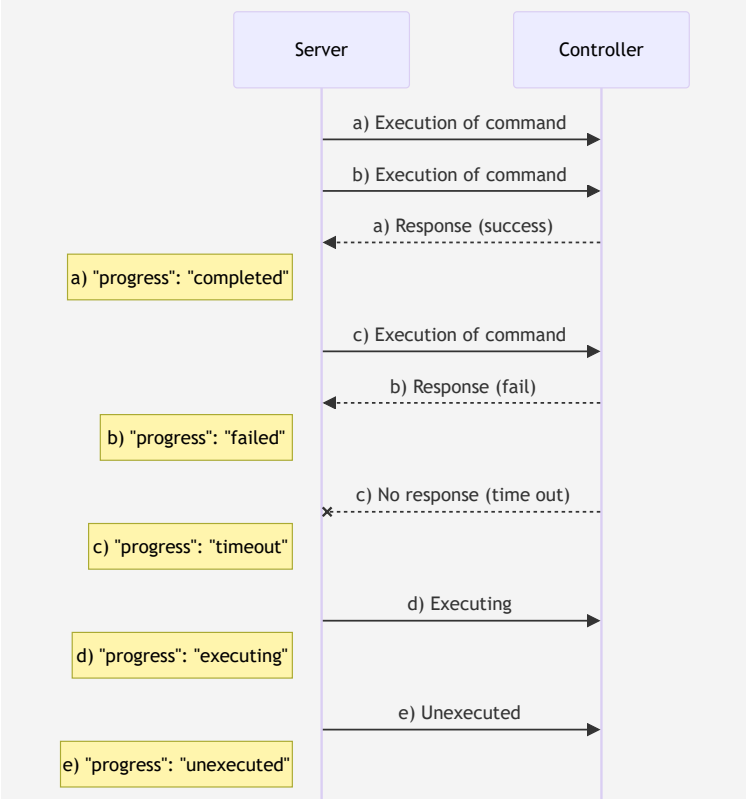


図 7-3　concurrentモード：d）まで実行中

″processMode″が″sequential″の場合は、各命令のサーバ実行が失敗するか（failedもしくはtimeout）、クライアントから明示的に abortされるか、サーバ側での処理時間が（サーバ規定の）全体タイムアウト値を超えない限り、全ての命令が実行される。すなわち、正常応答（succeeded）が続く限り、全ての命令が実行される。

図 7-4に″sequential″モードでの処理について例示する。図では、命令a）から e）を含んだbulkの各命令を順次実行中であり、a）b）は成功し、c）以降は未実行の状態を示している（クライアントが呼び出した時点は b)直後の位置）。

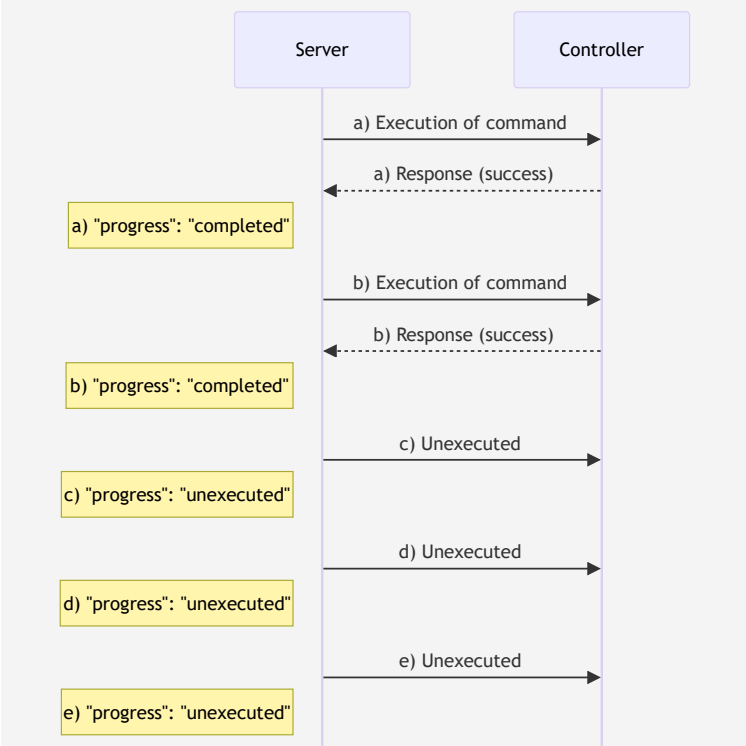


図 7-4　sequentialモード：b）まで実行中

続いて、図 7-5では、c）が失敗に終わったため、d）e）は未実行から中断状態へ移行したことを示している。c）で失敗しているため、以降の命令実行は行われない。

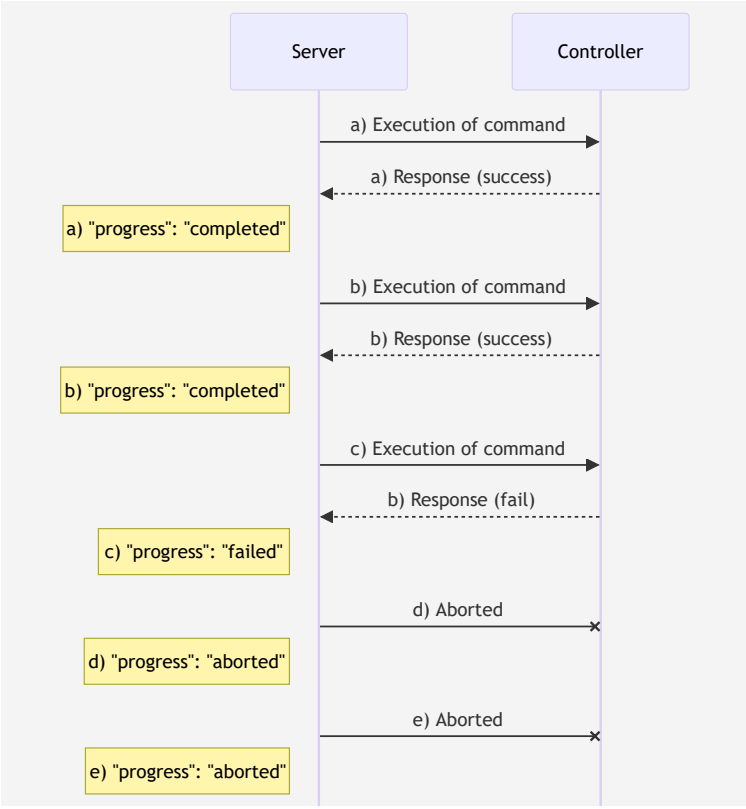


図 7-5　sequentialモード：c）実行後停止

"processMode"はいずれの場合も、クライアントから abortが送信された場合は、サーバ上にて指定されたexecution IDに対応する残存の実行処理を中断し、"progress"が実行中（executing）および未実行（unexecuted）の命令は中断（aborted）へ移行する。bulk実行有効期間は、前述の通りサーバ上で一定時間後に自動的に削除されない限り継続するため、クライアントは中断時の状況についてexecution IDを伴う getResultsアクションにて取得することが可能である。

⑪のレスポンス内に列挙される各命令の"progress"を全て確認すれば、全体の動作状況を把握することは可能だが、対象命令数が多数となる場合、クライアントの状況確認処理が煩雑となってしまう。クライアントが簡易にbulk実行に関する全体の進行状況を把握できるよう、"processStatus"が提供されている。

"processStatus"は、"inProgress"(実行中)，"succeeded"(すべて成功にて実行完了)，"failed"(1つ以上の失敗またはタイムアウトにて実行完了)，"aborted"(1つでも中断がある場合。"failed"より優先)のいずれかの値をとる。

全体の進行状況に応じた"processStatus"の値については下記の通りとなる。

- 全ての命令が未実行または1つ以上の命令が実行中の場合：inProgress"
- bulk実行途中でクライアントから abort発行にて中断された場合、サーバシステム全体でのタイムアウト値を超えて中断された場合、"processMode"が"sequential"モードにて"failed"または"timeout"が発生した場合のいずれか："aborted"
- 全ての命令実行が完了した場合：
  - 全ての実行が成功→"succeeded"
  - 1つ以上の実行が失敗またはタイムアウトした場合→"failed" タイムアウト値の設定は、サーバにて適時規定されていることを前提する。各命令の実行タイムアウト値t_i（iは1からtotal）や、全命令の実行タイムアウト値T（bulk実行有効期間）については、必ずしも各実行タイムアウト値の総和（$\sum_{i=1}^{n} t_i$）以上である必要はないが、前述の通り、十分余裕を持った期間に設定されることが望ましい。 表 7-1に、bulkに関するAPI一覧を示す。以降、個別に説明する。

表 7-1　複数命令の一括指示に関するAPI

http method	path	description
POST	/bulks	bulk作成
GET	/bulks	bulk IDの一覧取得
GET	/bulks/<bulk id>	bulk description取得
GET	/bulks/<bulk id>/properties	bulkの全プロパティリソース値の取得
GET	/bulks/<bulk id>/properties/<property resource name>	bulkのプロパティリソース値の取得
PUT	/bulks/<bulk id>/properties/<property resource name>	bulkのプロパティリソース値の設定
POST	/bulks/<bulk id>/actions/execute	bulk実行開始

http method	path	description
POST	/bulks/<bulk id>/actions/abort	bulk実行中断
POST	/bulks/<bulk id>/actions/getResults	bulk実行結果の取得
DELETE	/bulks/<bulk id>	bulk削除

POST /bulks

bulkを作成・登録し、識別子（bulk ID）を生成しクライアントへ返却する。

対象リソース（命令）の共通URI部を"base"（省略可能）にて指定し、bulk処理モード（"processMode"）の指定や、各命令（"requests"の配列形式）を列挙してクライアントからサーバへ要求する。"processMode"は省略可能であり、省略時は、"concurrent"が適用される。各命令は、"requests"キー内の配列に列挙される各命令"method"(HTTPリクエストメソッド)、"path"("base"からの相対リソース位置。"base "が無指定の場合は絶対リソース位置), ["body"(引数となるHTTPリクエストボディ。省略可)]の組から構成される。サーバ上でのbulk作成・登録が成功すると、サーバからクライアントへHTTPステータスコード201とHTTPボディとしてbulk IDが返却される。サーバにて登録可能な最大bulk数を超える場合は、HTTPステータスコード400とHTTPボディとして { "type": "rangeError", "message": "You can't create bulks over the registration limit" }が返却される。

■リクエスト定義

```
{
  "descriptions": {
    "ja": <description in Japanese>,
    "en": <description in English>
  },
  "base": <base uri>,
  "processMode": "concurrent" | "sequential",
  "requests": [
    {
      "method": <http method>,
      "path": <path>,
      "body": <value>(optional)
    },
    ...
  ]
}
```

■レスポンス定義

```
{
  "id": <bulk id>
```

```
}
```

■リクエスト例

```
{
    "descriptions": {
        "ja": "帰宅",
        "en": "I'm home"
    },
    "base": "https://xxx.xxx/elapi/v1/",
    "processMode": "concurrent",
    "requests": [
        {
            "method": "GET",
            "path": "devices/0123/properties/operationStatus"
        },
        {
            "method": "PUT",
            "path": "devices/0124/properties/targetTemperature",
            "body": {
                "targetTemperature": 24
            }
        }
    ]
}
```

■レスポンス例

```
{
  "id": "00000011"
}
```

リクエスト：

Property	Type	Required	Description
descriptions	object	Yes	登録するbulkに関する記述
descriptions.ja	string	Yes	登録するbulkの内容説明（日本語）
descriptions.en	string	Yes	登録するbulkの内容説明（英語）
base	string	No	操作対象となるリソースへの共通ベースURI（省略可）

Property	Type	Required	Description
processMode	string	No	"concurrent"または"sequential"。省略時には"concurrent"モードとなる。前者はrequests内で登録する命令セットを並列実行し、後者は逐次実行する。並列実行では、各命令を記述順に順次実行するが、各応答は待たずに実行される。逐次実行では、命令の正常応答を待ってから、次の命令を実行する
requests	array	Yes	各命令を配列にて列挙
requests[].method	string	Yes	HTTP リクエストメソッド
requests[].path	string	Yes	操作対象とするリソースへのパス。baseが指定された場合は、相対パスにて指定し、省略された場合は絶対パス（URI）にて指定
requests[].body	object	No	操作時に必要なリクエストボディ

レスポンス：

Property	Type	Required	Description
id	string	Yes	bulk ID

GET /bulks

builk IDの一覧取得。サーバに登録されたbulk IDが配列形式にて返却される。登録が無い場合は、空の配列が返却される。

■レスポンス定義

```
{
    "registrationLimit": <maximum number of registered bulks>,
    "bulks": [
        {
            "id": <bulk id>,
            "descriptions": {
                "ja": <description in Japanese>,
                "en": <description in English>
            }
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "registrationLimit": 100,
    "bulks": [
        {
            "id": "00000011",
            "descriptions": {
                "ja": "帰宅",
                "en": "I'm home"
            }
        },
        {
            "id": "00000012",
            "descriptions": {
                "ja": "外出",
                "en": "I'm out"
            }
        }
    ]
}
```

レスポンス：

Property	Type	Required	Description
registrationLimit	number	No	登録可能なbulkの最大数
bulks	array	Yes	登録済のbulk ID等を配列にて列挙。登録が無い場合は空（"bulks": []）
bulks[].id	string	No	bulk ID
bulks[].descriptions	object	No	登録済のbulkに関する記述
bulks[].descriptions.ja	string	No	登録済のbulkの内容説明（日本語）
bulks[].descriptions.en	string	No	登録済のbulkの内容説明（英語）

GET /bulks/<bulk id>

bulk IDで指定されたbulk descriptionを取得する。

■レスポンス例

```
{
    "properties": {
        "descriptions": {
            "descriptions": {
                "ja": "bulkの説明",
                "en": "explanation of bulk"
```

```
            },
            "writable": true,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "ja": {
                        "type": "string"
                    },
                    "en": {
                        "type": "string"
                    }
                }
            }
        },
        "base": {
            "descriptions": {
                "ja": "ベースのURI",
                "en": "base URI"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
        "processMode": {
            "descriptions": {
                "ja": "処理モード",
                "en": "processing mode"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
                    "concurrent",
                    "sequential"
                ]
            }
        },
        "requests": {
            "descriptions": {
                "ja": "bulkの中の要求命令セット",
                "en": "set of request commands in a bulk"
            },
            "writable": false,
            "observable": false,
            "schema": {
```

```
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "method": {
                            "type": "string",
                            "enum": [
                                "GET",
                                "PUT",
                                "POST",
                                "PATCH",
                                "DELETE"
                            ]
                        },
                        "path": {
                            "type": "string"
                        },
                        "body": {
                            "type": "object"
                        }
                    }
                }
            }
        }
    },
    "actions": {
        "execute": {
            "descriptions": {
                "ja": "bulkを実行",
                "en": "Execute a bulk"
            },
            "schema": {
                "type": "object",
                "properties": {
                    "executionId": {
                        "type": "string"
                    }
                }
            }
        },
        "abort": {
            "descriptions": {
                "ja": "bulk実行中断",
                "en": "Abort an execution of a bulk"
            },
            "input": {
                "type": "object",
                "properties": {
                    "executionId": {
```

```
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "executionId": {
                        "type": "string"
                    }
                }
            }
        },
        "getResults": {
            "descriptions": {
                "ja": "実行結果の取得",
                "en": "Get results"
            },
            "input": {
                "type": "object",
                "properties": {
                    "executionId": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "base": {
                        "type": "string"
                    },
                    "total": {
                        "type": "number",
                        "minimum": 0
                    },
                    "processStatus": {
                        "type": "string",
                        "enum": [
                            "succeeded",
                            "failed",
                            "inProgress",
                            "aborted"
                        ]
                    },
                    "responses": {
                        "type": "array",
                        "items": {
                            "type": "object",
```

```
                        "properties": {
                            "method": {
                                "type": "string",
                                "enum": [
                                    "GET",
                                    "PUT",
                                    "POST",
                                    "PATCH",
                                    "DELETE"
                                ]
                            },
                            "path": {
                                "type": "string"
                            },
                            "body": {
                                "type": "object"
                            },
                            "progress": {
                                "type": "string",
                                "enum": [
                                    "unexecuted",
                                    "executing",
                                    "completed",
                                    "failed",
                                    "timeout",
                                    "aborted"
                                ]
                            },
                            "statusCode": {
                                "type": "number"
                            }
                        }
                    }
                }
            }
        }
    }
}
```

レスポンス：

Property	Type	Required	Description
properties	object	Yes	「GET /bulks/<bulk id>/properties」におけるプロパティリソースに関する定義`

Property	Type	Required	Description
actions	object	Yes	「POST /bulks/<bulk id>/actions/[execute, abort, getResults]」におけるアクションリソースに関する記述

GET /bulks/<bulk id>/properties

bulk IDで指定された全プロパティリソースの値を取得する。

■レスポンス定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
    "base": <base uri>,
    "processMode": "concurrent"|"sequential",
    "requests": [
        {
            "method": <http method>,
            "path": <path>,
            "body": <value>(optional)
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "descriptions":{
        "ja": "帰宅",
        "en": "I'm home"
    },
    "base": "https://xxx.xxx/elapi/v1/",
    "processMode": "concurrent",
    "requests": [
        {
            "method": "GET",
            "path": "devices/0123/properties/operationStatus"
        },
        {
            "method": "PUT",
            "path": "devices/0124/properties/targetTemperature",
            "body": {
```

```
                "targetTemperature": 24
            }
        }
    ]
}
```

レスポンス：

Property	Type	Required	Description
descriptions	object	Yes	登録済のbulkに関する記述
descriptions.ja	string	Yes	登録済のbulkの内容説明（日本語）
descriptions.en	string	Yes	登録済のbulkの内容説明（英語）
base	string	No	操作対象となるリソースへの共通ベースURI（bulk生成時に無指定の場合は、省略）
processMode	string	Yes	"concurrent"または"sequential"。bulk生成時に無指定の場合は"concurrent"モードとなる。前者は requests内で登録する命令セットを並列実行し、後者は逐次実行する。並列実行では、各命令を記述順に順次実行するが、各応答は待たずに実行される。逐次実行では、命令の正常応答を待ってから、次の命令を実行する
requests	array	Yes	各命令を配列にて列挙
requests[].method	string	Yes	HTTP リクエストメソッド
requests[].path	string	Yes	操作対象とするリソースへのパス。base以下のパス。baseが存在する場合は相対パス、省略された場合は絶対パス（URI）
requests[].body	object	No	操作時に必要なリクエストボディ。requests[].methodがGETなど、不要な場合は省略される

GET /bulks/<bulk id>/properties/<property resource name>

bulk IDにて指定されたbulkのプロパティリソース値の取得。

■リクエスト例

```
GET /bulks/00000012/properties/processMode
```

■レスポンス定義

```
{
    <property resource name>: <property value>
}
```

■レスポンス例

```
{
    "processMode": "concurrent"
}
```

PUT /bulks/<bulk id>/properties/<property resource name>

bulk IDにて指定されたbulkのプロパティリソース値の設定。下記例では、descriptions部の書き換えが可能なことを示している。

■リクエスト定義およびレスポンス定義

```
{
    <property resource name>: <property value>
}
```

■リクエスト例およびレスポンス例

```
{
    "descriptions": {{"ja": "就寝"}, {"en": "I'm going to sleep"}}
}
```

POST /bulks/ <bulk id>/actions/execute

bulk実行の開始。リクエスト時、ボディの指定は不要。bulkの実行では、全てのリクエストの実行が完了するまでに時間がかかることを想定し、実行結果については非同期で取得するモデルを採用している。具体的には、クライアントは POST actions/executeでbulk実行の開始を指示しexecution IDを取得し、このIDを指定して POST actions/getResultsを呼び出すことで結果を取得する。

execution IDはtemporaryなidのため、検索や削除の手段は提供していない。サーバが一定時間後に削除する実装とする（execution IDの有効期限はサーバ実装依存とする）。processStatus 完了なら次受付。でなければエラー。

■レスポンス定義

```
{
    "executionId": <execution id>
}
```

■レスポンス例

```
{
    "executionId": "0023"
}
```

レスポンス：

Property	Type	Required	Description
executionId	string	Yes	実行時に割り当てられるexecution ID

POST /bulks/<bulk id>/actions/abort

bulk実行の中断。リクエスト時、ボディにて中断対象となるexecution IDを指定する。bulk実行の中断に成功した場合は、同execution IDを返す。

■リクエスト定義

```
{
    "executionId": <execution id>
}
```

■リクエスト例

```
{
    "executionId": "0023"
}
```

■レスポンス定義

```
{
    "executionId": <execution id>
}
```

■レスポンス例

```
{
    "executionId": "0023"
}
```

レスポンス：

Property	Type	Required	Description
executionId	string	Yes	実行時に割り当てられるexecution ID

POST /bulks/<bulk id>/actions/getResults

指定したexecution IDに対応するbulk実行のレスポンス取得。

■リクエスト定義

```
{
    "executionId": <execution id>
}
```

■リクエスト例

```
{
    "executionId": "0023"
}
```

■レスポンス定義：

```
{
    "base": <base uri>,
    "total": <total number of the requests>,
    "processStatus": <processing status of whole commands>,
    "responses": [
        {
            "method": <http method>,
            "path": <path>,
            "body": <body data>,
            "progress": <progress state>,
            "status": <status code>
        },
        ...
    ]
}
```

■レスポンス例：


```
{
    "base": "https://xxx.xxx/elapi/v1/",
    "total": 3,
    "processStatus": "inProgress",
    "responses": [
        {
            "method": "GET",
            "path": "devices/0123/properties/operationStatus",
            "body": {
                "operationStatus": true
             },
            "progress": "completed","status": 200
        },
        {
            "method": "PUT",
            "path": "devices/0124/properties/targetTemperature",
            "body": {
                "type": "rangeError",
                "message": "..."
            },
            "progress": "failed",
            "status": 400
        },
        {
            "method": "GET",
            "path": "devices/0124/properties/roomTemperature",
            "progress": "unexecuted"
        }
    ]
}
```


レスポンス：

Property	Type	Required	Description
base	string	No	操作対象となるリソースへの共通ベースURI（bulk生成時に無指定の場合は、省略）
total	number	Yes	実行命令のトータル数

Property	Type	Required	Description
processStatus	string	Yes	全体の進行状況。"inProgress"(実行中), "succeeded"(すべて成功にて実行完了), "failed"(1つ以上の失敗またはタイムアウトにて実行完了), "aborted"(1つでも中断がある場合。"failed"より優先)のいずれかの値
responses	array	Yes	各命令の応答を配列にて列挙
responses[].method	string	Yes	HTTP リクエストメソッド
responses[].path	string	Yes	操作対象とするリソースへのパス。base以下のパス。baseが存在する場合は相対パス、省略された場合は絶対パス（URI）
responses[].body	object	No	操作実行成功後には応答（レスポンスボディ）。実行エラー時にはエラー詳細を記述する場合は"error"と"message"のオブジェクト形式（オプション）。また、操作未実行時や実行中の場合はリクエストボディと同じ内容（requests[].methodがGETで、未実行・実行中の場合は省略される）
responses[].progress	string	Yes	操作の実行状況。操作未実行時は"unexecuted"、実行中は"executing"、操作実行後、成功完了時には "completed"、失敗完了時には"failed"、タイムアウト時には"timeout"、実行中断は"aborted"のいずれかの値
responses[].status	string	No	操作実行後の応答（ステータスコード）。操作未実行時、実行中は省略される

DELETE /bulks/<bulk id>

登録済みのbulkの削除。bulk IDには、削除対象となるbulk IDを指定する。

Bodyは含まず、HTTPステータスコード204(No Content)のみ返す。 既に削除済みもしくは存在しないbulk IDが指定された場合は、404(Not Found)を返す。

7.2 機器のグルーピング（groups）

サーバは、クライアントからの要求に基づき、複数の機器をまとめてグループ化し登録することで、リソースの分類・整理や対象機器群に対する個別・共通命令などを実行可能にする機能（group操作機能）をクライアントに対して提供する。

また、クライアントからのリクエストがなくても、サーバはgroupを作成することを可能とする。例えば「複数の機器オブジェクトで構成された製品の場合」や「家庭内の HEMS コントローラが複数の製品をグループ化した場合」などに、サーバが必要に応じて group をあらかじめ登録することなどが考えられる。サーバが登録した group か、それともクライアントの要求で作成された

group であるかを区別する手段として、 preConfigured というプロパティを定義する。 図 7-6 を用いて、group操作に関する一連の動作について例示する。

- group操作のためには、まず、①クライアントが、グループ化対象となる機器を複数列挙した機器リスト（group）をサーバへ登録申請する。サーバは、②同groupを登録後、③クライアントへ同groupの識別子となるgroup IDを返却する。
- 次に、④⑦⑫のうちいずれかを選択し、所望のgroup操作を行うことができる。
- ④では、クライアントから同group IDを指定してグループ化された機器への全プロパティリソース値の取得、⑦では、指定プロパティリソースに対する値の取得、⑫では、指定プロパティリソースに対する値の設定が可能となる。
- 最終的に、登録したgroupが不要となった場合は、group IDを指定してサーバ登録されたgroupを消去する（⑰⑱⑲）。

デフォルトでは、④⑦⑫のようなプロパティリソース操作を実施する際、対象機器が当該プロパティを有し実行可能なメソッドであれば実施され、当該プロパティを有しない場合やメソッドに対応していない場合はHTTPステータスコード404（Not Found）や405（Method Not Allowed）を伴い、エラーメッセージを返却する。プロパティ操作が実行された場合であっても、実機からの応答時間がサーバ規定のタイムアウト値を超える場合は、HTTPステータスコード500（Internal Server Error）を伴い、エラーメッセージ（"type": "timeoutError"）を返却する。

生成するグループの特質によっては、上記のようなエラーケースで400系、500系コードやエラーメッセージの返却をクライアントが不要としたい場合がある。例えば、1台以上の蓄電池クラスと1台以上の住宅用太陽光発電クラスを組み合わせたグループを作成し、機器全体を仮想的な混合デバイス（以降、「仮想混合デバイス」と呼ぶ）として扱いたい場合、2つのクラス共通のプロパティを操作する場合は、全ての機器に対して操作を実施し、一方のクラスのみが搭載するプロパティを操作する場合は、不要な他方のクラス（機器）へ操作を行わない（エラー処理を省略する）処理を期待するケースが考えられる。オプション機能として、こうした処理を可能にする仕組みを提供する。

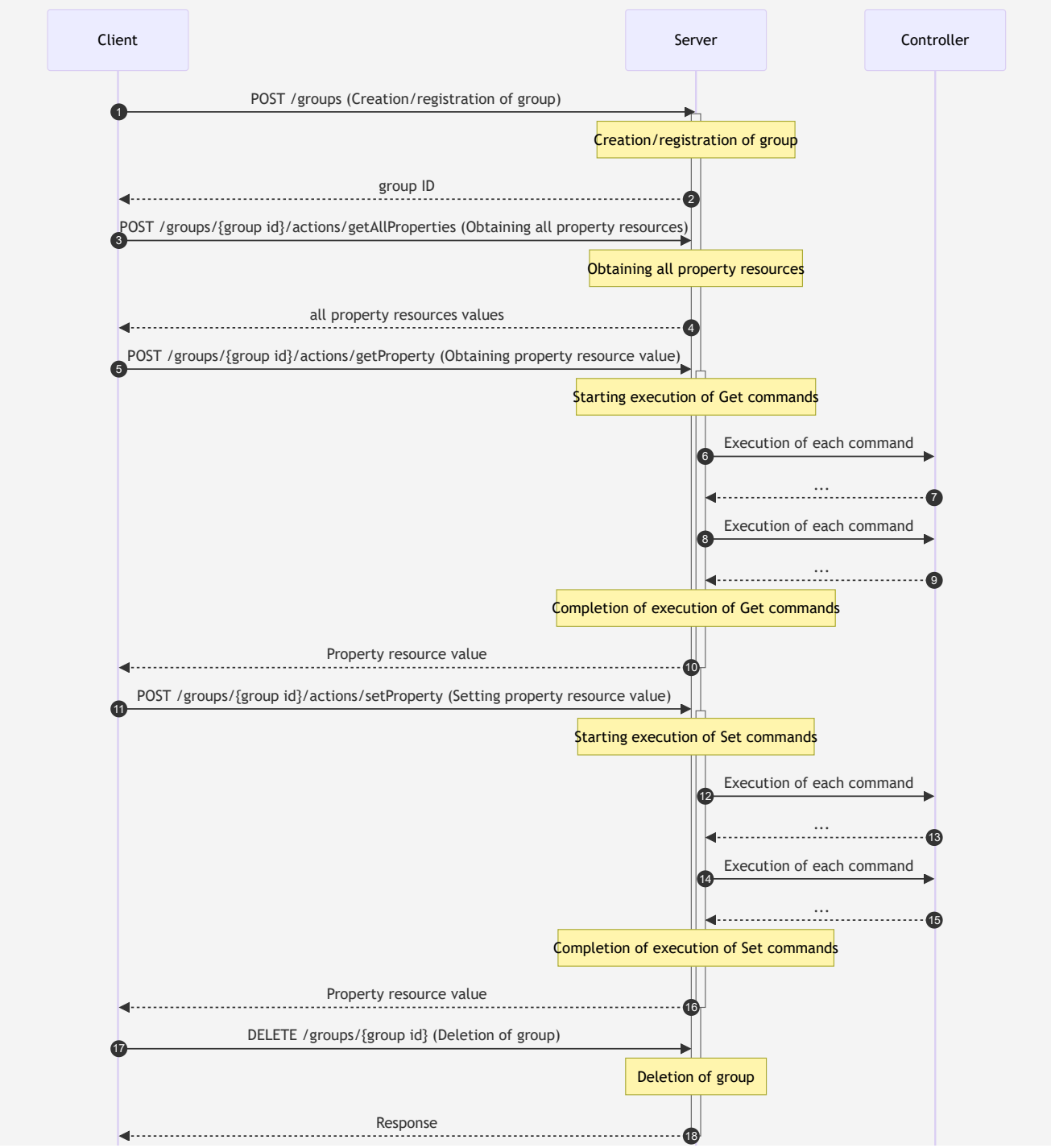


図 7-6　group実行の全体概要

なお、明示的に操作対象となるクラスに対応した機器を指定したい場合は、下記のような指定形式も可能とする（オプション）。

例）仮想混合デバイスのうち一般照明クラス（generalLighting）対応機器のみを対象とした操作

```
POST /groups/<group id>/actions/getProperty?deviceType=generalLighting
HTTP/1.1
```

また、一旦グループを作成した後に、グループ化対象となる機器の追加・削除機能も提供する。

表 7-2に、groupに関するAPI一覧を示す。以降、個別に説明する。

表 7-2　機器のグルーピングに関するAPI

http method	path	description
POST	/groups	group作成
GET	/groups	group IDの一覧取得
GET	/groups /<group id>	group description取得
GET	/groups /<group id>/properties	groupの全プロパティリソース値の取得
GET	/groups/<group id>/properties/<property resource name>	groupのプロパティリソース値の取得
PUT	/groups/<group id>/properties/<property resource name>	groupのプロパティリソース値の設定
POST	/groups /<group id>/actions/getAllProperties	対象機器群の全プロパティリソース値の取得
POST	/groups /<group id>/actions/getProperty	対象機器群のプロパティ値の取得
POST	/groups /<group id>/actions/setProperty	対象機器群のプロパティ値の設定
DELETE	/groups /<group id>	group削除

POST /groups

グループ化したい対象機器を保持するグループ（group）を作成・登録し、識別子（group ID）を生成しクライアントへ返却する。 対象機器はdevice IDの配列形式にて列挙してクライアントからサーバへ要求する。サーバにて登録可能な最大group数を超える場合は、HTTPステータスコード400とHTTPボディとして { "type": "rangeError", "message": "You can't create groups over the registration limit" }が返却される。

以降、デフォルトでは、グループ化された機器全てに対して、一括で処理が可能となる（オプションである"composed"を false指定した場合も同様）。

仮想混合デバイスがサポートされている場合は、"composed"に trueを指定することで、以降、グループ対象機器群のプロパティに対する操作は、当該プロパティを有し実行可能なメソッドに対応している機器に対してのみ実行され、そうでない機器に対しては当該操作が実行されず、レスポンスも返却されない（エラーコードやエラーメッセージを返さない）。

"composed"がサポートされていないサーバに対して、"composed"に trueを指定してリクエストした場合、group IDは返却されず、HTTPステータスコード404 (Not Found) を伴い、エラーメッセ

ージ（"type": "typeError"）が返却される。

■リクエスト定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
    "members": [
        {
            "deviceId": <device id>
        },
        ...
    ],
    "composed": <true/false>
}
```

■リクエスト例

```
{
    "descriptions": {
        "ja": "リビング",
        "en": "living"
    },
    "members": [
        {
            "deviceId": "0123"
        },
        {
            "deviceId": "1234"
        },
        {
            "deviceId": "2345"
        }
    ],
    "composed": true
}
```

■レスポンス定義

```
{
    "id": <group id>
}
```

■レスポンス例

```
{
    "id": "00000011"
}
```

リクエスト：

Property	Type	Required	Description
descriptions	object	Yes	登録するgroupに関する記述
descriptions.ja	string	Yes	登録するgroupの内容説明（日本語）
descriptions.en	string	Yes	登録するgroupの内容説明（英語）
members	array	Yes	対象機器群。device IDを配列にて列挙
members[].deviceId	string	Yes	対象機器のdevice ID
composed	boolean	No	仮想混合デバイスのサポートの有無。省略された場合はfalseとなる。

レスポンス：

Property	Type	Required	Description
id	string	Yes	group ID

GET /groups

group IDの一覧取得。サーバに登録されたgroup IDが配列形式にて返却される。登録が無い場合は、空の配列が返却される。

■レスポンス定義

```
{
    "registrationLimit": <maximum number of registered groups>,
    "groups": [
        {
            "id": <group id>,
            "descriptions": {
                "ja": <description in Japanese>,
                "en": <description in English>
            }
        },
        ...
```

```
    ]
}
```

■レスポンス例

```
{
    "registrationLimit": 100,
    "groups": [
        {
            "id": "00000011",
            "descriptions": {
                "ja": "リビング",
                "en": "living"
            }
        },
        {
            "id": "00000012",
            "descriptions": {
                "ja": "寝室",
                "en": "bedroom"
            }
        }
    ]
}
```

レスポンス：

Property	Type	Required	Description
registrationLimit	number	No	登録可能なgroup の最大数
groups	array	Yes	登録済のgroup ID等を配列にて列挙。登録が無い場合は空（"groups": []）
groups[].id	string	No	group ID
groups[].descriptions	object	No*1	登録済のgroupに関する記述
groups[].descriptions.ja	string	No*1	登録済のgroupの内容説明（日本語）
groups[].descriptions.en	string	No*1	登録済のgroupの内容説明（英語）

*1) groups[].id が存在する場合に必須

GET /groups/<group id>

group IDで指定されたgroup descriptionを取得する。

■レスポンス例

```
{
    "properties": {
        "descriptions": {
            "descriptions": {
                "ja": "groupの説明",
                "en": "explanation of group"
            },
            "writable": true,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "ja": {
                        "type": "string"
                    },
                    "en": {
                        "type": "string"
                    }
                }
            }
        },
        "members": {
            "descriptions": {
                "ja": "groupに属する機器のdevice idのリスト",
                "en": "list of device ids in this group"
            },
            "writable": true,
            "observable": false,
            "schema": {
                "type": "array",
                "items": {
                    "type": "object",
                    "properties": {
                        "deviceId": {
                            "type": "string"
                        }
                    }
                }
            }
        },
        "composed": {
            "descriptions": {
                "ja": "仮想混合デバイスの設定",
                "en": "setting of virtual compound device"
            },
            "writable": false,
            "observable": false,
            "schema": {
```

```
                    "type": "boolean"
                }
            },
            "preConfigured": {
                "descriptions": {
                    "ja": "サーバ事前登録によるGroupか否か",
                    "en": "setting of server preconfigration"
                },
                "writable": false,
                "observable": false,
                "schema": {
                    "type": "boolean"
                }
            }
        },
        "actions": {
            "getAllProperties": {
                "descriptions": {
                    "ja": "グループに存在する機器の全プロパティを読み出す",
                    "en": "Read all properties of all devices in this group"
                },
                "schema": {
                    "type": "object",
                    "properties": {
                        "responses": {
                            "type": "array",
                            "items": {
                                "type": "object",
                                "properties": {
                                    "deviceId": {
                                        "type": "string"
                                    },
                                    "properties": {
                                        "type": "object"
                                    }
                                }
                            }
                        }
                    }
                }
            },
            "getProperty": {
                "descriptions": {
                    "ja": "グループに存在する機器の指定プロパティ値の取得",
                    "en": "Read the specified property of all devices in this
group"
                },
                "input": {
                    "type": "object",
```

```
                "properties": {
                    "propertyName": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "responses": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "deviceId": {
                                    "type": "string"
                                },
                                "body": {
                                    "type": "object"
                                },
                                "statusCode": {
                                    "type": "number"
                                }
                            }
                        }
                    }
                }
            }
        },
        "setProperty": {
            "descriptions": {
                "ja": "グループに存在する機器の指定プロパティ値の設定",
                "en": "Write the specified property of all devices in this
group"
            },
            "input": {
                "type": "object",
                "properties": {
                    "propertyName": {
                        "type": "string"
                    },
                    "propertyValue": {
                        "type": "object"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
```

```
                    "responses": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "deviceId": {
                                    "type": "string"
                                },
                                "body": {
                                    "type": "object"
                                },
                                "statusCode": {
                                    "type": "number"
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

レスポンス：

Property	Type	Required	Description
properties	object	Yes	「GET /groups/<group id>/properties」におけるプロパティリソースに関する定義
actions	object	Yes	「POST /groups/<group id>/actions/ [getAllProperties, getProperty, setProperty]」におけるアクションリソースに関する定義

GET /groups/<group id>/properties

group IDで指定されたgroupの全プロパティリソースの値を取得する。

■レスポンス定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
    "members": [{"deviceId": <device id>},... ],
    "composed": <true/false>,
```

```
    "preConfigured": <true/false>
}
```

■レスポンス例

```
{
    "descriptions": {"ja": "リビング", "en": "living"},
    "members": [{"deviceId": "0123"}, {"deviceId": "1234"}, {"deviceId":
"2345"}],
    "composed": true,
    "preConfigured": false
}
```

レスポンス：

Property	Type	Required	Description
descriptions	object	Yes	登録済のgroupに関する記述
descriptions.ja	string	Yes	登録済のgroupの内容説明（日本語）
descriptions.en	string	Yes	登録済のgroupの内容説明（英語）
members	array	Yes	対象機器のdevice IDを配列にて列挙
members[].deviceId	string	Yes	対象機器のdevice ID
composed	boolean	No	仮想混合デバイスのサポートの有無
preConfigured	boolean	No	サーバ事前登録groupか否か

GET /groups/<group id>/properties/<property resource name>

group IDにて指定されたgroupのプロパティリソース値の取得。

■リクエスト例

```
GET /groups/00000013/properties/members
```

■レスポンス定義

```
{
    <property resource name>: <property value>
}
```

■レスポンス例

```
{
    "members": [{"deviceId": "0123"}, {"deviceId": "1234"}, {"deviceId":
"2345"}]
}
```

PUT /groups/<group id>/properties/<property resource name>

group IDにて指定されたgroupのプロパティリソース値の設定。下記例では、機器リストの更新により対象とする機器群の入れ替えや増減が可能なことを示している。

■リクエスト定義およびレスポンス定義

```
{
    <property resource name>: <property value>
}
```

■リクエスト例およびレスポンス例

```
{
    "members": [{"deviceId": "1234"}, {"deviceId": "2345"}]
}
```

POST /groups/<group id>/actions/getAllProperties

group IDにて指定されたgroupが保持する対象機器群への全プロパティリソース値の取得。リクエスト時、ボディの指定は不要。

機器単体に対して <device ID>/propertiesを指定しGETメソッドを用いて取得される結果と同じ内容が"properties"キーに対応する値にて返却される。

■レスポンス定義

```
{
    "responses": [
        {
            "deviceId": <device id>,
            "properties": {
                <property resource name 1>: <value 1>,
                <property resource name 2>: <value 2>,
                ...
            }
        },
        ...
```

```
    ]
}
```

■レスポンス例

```
{
    "responses": [
        {
            "deviceId": "1234",
            "properties": {
                "operationStatus": true, ...
            }
        },
        {
            "deviceId": "2345",
            "properties": {
                "operationStatus": true, ...
            }
        },
    ]
}
```

レスポンス：

Property	Type	Required	Description
responses	array	Yes	レスポンス
responses[].deviceId	string	Yes	対象機器のdevice ID
responses[].properties	object	Yes	対象機器の全プロパティリソース値

POST /groups/<group id>/actions/getProperty

group IDにて指定されたgroupが保持する機器群へのプロパティリソース値の取得。

機器単体に対して <device ID>/properties/<property resource name>を指定しGETメソッドを用いて取得される結果と同じ内容が"body"キーに対応する値にて返却されると共に、HTTPステータスコードも返却される。エラーが発生する場合は、エラー値が返却されると共に、HTTPステータスコードも返却される。

"composed"がサポートされている場合（trueの場合）は、同property resource nameを有しGETメソッドに対応している場合のみ応答し、そうでない機器に関して操作を実行せず、結果も返却しない。

■リクエスト定義

```
{
    "propertyName": <property resource name>
}
```

■リクエスト例

```
{
    "propertyName": "operationStatus"
}
```

■レスポンス定義

```
{
    "responses": [
        {
            "deviceId": <device id>,
            "body": {
                <property resource name>: <value>
            },
            "status": <status code>
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "responses": [
        {
            "deviceId": "0123",
            "body":{
                "operationStatus": true
            },
            "status": 200
        },
        {
            "deviceId": "1234",
            "body":{
                "type": "timeoutError",
                "message": "the device does not respond"
            },
            "status": 500
```

```
        }
    ]
}
```

レスポンス：

Property	Type	Required	Description
responses	array	Yes	レスポンス
responses[].deviceId	string	Yes	device ID
responses[].body	object	Yes	操作実行成功後には応答（レスポンスボディ）。実行エラー時は"type"(必須)、"message"（必須）
responses[].status	number	Yes	操作実行後の応答（ステータスコード）

POST /groups/<group id>/actions/setProperty

group IDにて指定されたgroupが保持する機器群へのプロパティリソース値の設定。

機器単体に対して <device ID>/properties/<property resource name>を指定しSETメソッドを用いて取得される結果と同じ内容が"body"キーに対応する値にて返却されると共に、HTTPステータスコードも返却される。エラーが発生する場合は、エラー値が返却されると共に、HTTPステータスコードも返却される。

"composed"がサポートされている場合（trueの場合）は、同property resource nameを有しPUTメソッドに対応している場合のみ応答し、そうでない機器に関して操作を実行せず、結果も返却しない。

■リクエスト定義

```
{
    "propertyName": <property resource name>,
    "propertyValue": <property value>
}
```

■リクエスト例

```
{
    "propertyName": "operationStatus",
    "propertyValue": true
}
```

■レスポンス定義

```
{
    "responses": [
        {
            "deviceId": <device id>,
            "body": {
                <property resource name>: <value>
            },
            "status": <status code>
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "responses": [
        {
            "deviceId": "0123",
            "body":{
                "operationStatus": true
            },
            "status": 200
        },
        {
            "deviceId": "1234",
            "body":{
                "type": "timeoutError",
                "message": "the device does not respond"
            },
            "status": 500
        }
    ]
}
```

レスポンス：

Property	Type	Required	Description
responses	array	Yes	レスポンス
responses[].deviceId	string	Yes	device ID
responses[].body	object	Yes	操作実行成功後には応答（レスポンスボディ）。実行エラー時は"type"（必須）、"message"（必須）

Property	Type	Required	Description
responses[].status	string	Yes	操作実行後の応答（ステータスコード）

DELETE /groups/<group id>

登録済みのgroupの削除。group IDには、削除対象となるgroup IDを指定する。

ボディは含まず、HTTPステータスコード204(No Content)のみ返す。 既に削除済みもしくは存在しないgroup IDが指定された場合は、404(Not Found)を返す。

7.3 履歴データ（histories）

サーバは、対象機器の各リソースに関する取得値を取得時刻とともに履歴データとして蓄積し、クライアントからの要求に基づき必要な履歴データをクライアントに対して提供する。

履歴データは、取得リソース値と取得時刻の組から構成される。サーバは、履歴データ保存対象となる機器や履歴データの記録タイミング・記録期間などを含んだ履歴データ属性（history）に関する管理を自ら行う。今回のバージョンでは、クライアントから履歴データ保存対象となる機器や取得リソースをサーバへ指定・登録する手段は提供していない。クライアントに対しては、サーバが独自に規定・保存した対象や期間に基づき提供される履歴データセットについて検索手段を提供するのみとする。履歴データセットは、あるhistory IDに基づきサーバ上で随時保存される全期間の履歴データ集合となる。

図 7-7を用いて、history操作に関する一連の動作について例示する。

- クライアントはサーバに対して、①history IDの一覧を取得要求可能であり、②サーバから得られたhistory ID一覧の中から③特定のhistory IDを指定して同IDに紐付く履歴データの定義情報（history description）を取得できる（④）。
- 続いて、クライアントは⑤パス上にてhistory IDに続き"properties"を指定することで、サーバから同IDに紐付くhistoryの全リソースプロパティ値を取得できる（⑥）。
- クライアントはサーバに対して、⑦history IDを指定して（必要に応じて期間や件数を含めた）prepareRetrieveDataアクションを実行することで、サーバが提供する履歴データセットの中からクライアント所望の履歴データサブセットを返却できるよう準備開始を指示する。⑧サーバは、同準備が完了すると、このアクションに紐づくdata IDに加え、範囲確定された履歴データの総件数や1応答（ページ）あたりの件数をクライアントへ返却する（⑨）。
- 以降、⑩クライアントはdata IDとページ番号を指定し retrieveアクションを実行することで、サーバより所望の履歴データサブセットを取得する（⑪）。
- 対象となる履歴データの総件数が1応答あたりの件数よりも多い場合は、履歴データサブセットは分割されるため、⑫2ページ目以降の番号を指定することで後続の履歴データサブセットを取得でき（⑬）、⑭終了ページ番号目（（総件数）／（1応答あたりの件数）：小数点以下切り上げ）にて最終となる履歴データサブセットを取得できる。
- 履歴データサブセットの有効期間は、サーバにより自由に規定可能とするが、通常は、クライアントによる retrieveアクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。

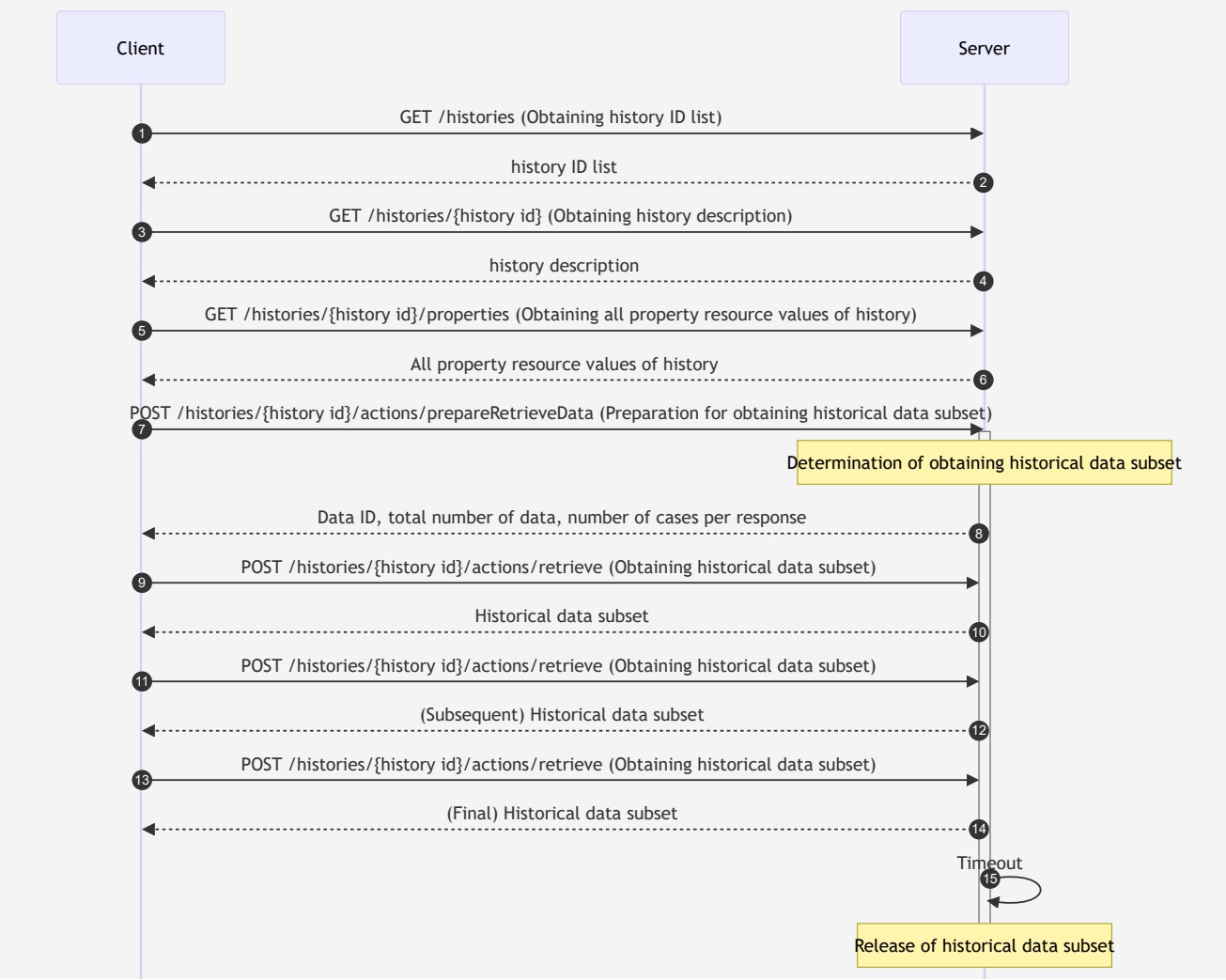


図 7-7　histories実行の全体概要

図 7-8に、サーバ規定による履歴データセットと、クライアント指定による履歴データサブセットとの関係について説明する。図は、8時から記録を開始し、30分毎に履歴データの保存を継続中で、現在時刻は11時45分頃を示している。履歴データセットは8時から11時30分までの履歴データ集合である。これに対し、9時から10時までの履歴データ集合は、クライアントから指定された履歴データサブセットとなる。クライアントから指定される期間の内容によっては、履歴データセットと履歴データサブセットが一致する場合もある。

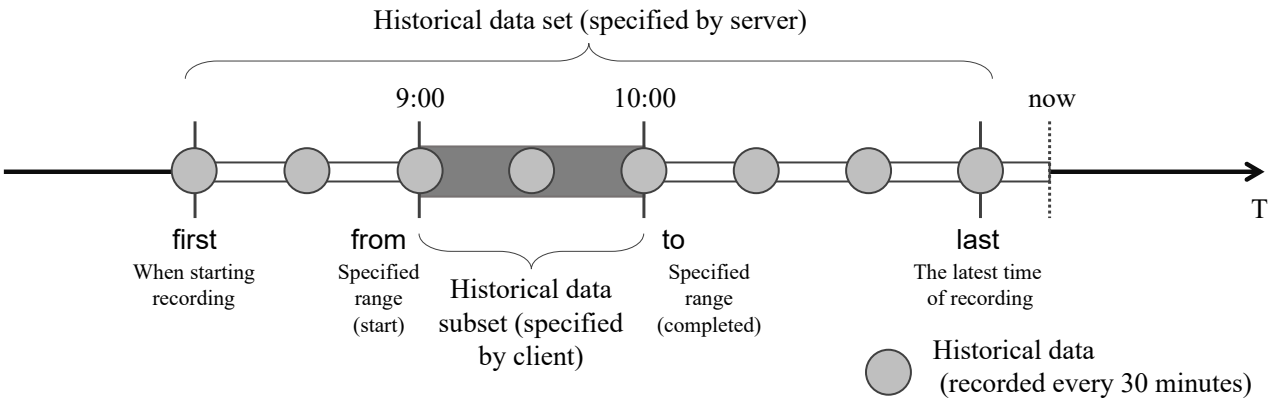


図 7-8　履歴データセット・サブセットの関係

表 7-3に、histories関するAPI一覧を示す。以降、個別に説明する。

表 7-3　履歴データに関するAPI

http method	path	description
GET	/histories	history IDの一覧取得
GET	/histories/<history id>	history description取得
GET	/histories/<history id>/properties	historyの全プロパティリソース値の取得
GET	/histories/<history id>/properties/<property resource name>	historyのプロパティリソース値の取得
POST	/histories/<history id>/actions/prepareRetrieveData	履歴データサブセット取得の準備指示
POST	/histories/<history id>/actions/retrieve	履歴データサブセットの取得

*) なお、historyのプロパティリソース値の設定は、historyにて書き込み可能なプロパティリソースが存在しないため省略。

GET /histories

履歴IDの一覧取得。サーバ登録されている履歴IDが配列形式にて返却される。登録が無い場合は、空の配列が返却される。

■レスポンス定義

```
{
    "histories": [
        {
            "id": <history id>,
            "descriptions": {
                "ja": <description in Japanese>,
                "en": <description in English>
            }
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "histories": [
        {
            "id": "00000011",
            "descriptions": {
                "ja": "照明000023の動作状態履歴",
                "en": "Operation status history of light 000023"
            }
        },
        {
            "id": "00000012",
            "descriptions": {
                "ja": "太陽光発電（PV）000045の積算発電電力量計測値（kWh）",
                "en": "Cumulative amount of electric energy generated of
PV 000045 ([kWh], 30min interval) "
            }
        }
    ]
}
```


レスポンス：

Property	Type	Required	Description
histories	array	Yes	それぞれの履歴データに対応する id, descriptions を property とする object を列挙する。複数機器履歴データ が存在しない場合は、要素が空の array をレスポンスする。
histories[].id	string	No	history ID
histories[].descriptions	object	No*1	登録済のhistoryに関する記述
histories[].descriptions.ja	string	No*1	登録済のhistoryの内容説明（日本語）
histories[].descriptions.en	string	No*1	登録済のhistoryの内容説明（英語）

*1) histories[].id が存在する場合に必須

GET /histories/<history id>

history idで指定されたhistory descriptionを取得する。

■レスポンス例


```
{
    "properties": {
```

```
        "descriptions": {
            "descriptions": {
                "ja": "historyの説明",
                "en": "explanation of history"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "ja": {
                        "type": "string"
                    },
                    "en": {
                        "type": "string"
                    }
                }
            }
        },
        "deviceId": {
            "descriptions": {
                "ja": "履歴データ記録対象機器のdevice ID",
                "en": "device ID of the device for which data are
recorded"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
        "deviceType": {
            "descriptions": {
                "ja": "履歴データ記録対象機器のdeviceType",
                "en": "deviceType of the device for which data are
recorded"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
        "resourceType": {
            "descriptions": {
                "ja": "リソースタイプ(property, action, event)",
                "en": "resource type (property, action, event)"
            },
            "writable": false,
```

```
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
                    "property",
                    "action",
                    "event"
                ]
            }
        },
        "resourceName": {
            "descriptions": {
                "ja": "リソース名",
                "en": "resource name"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
        "timing": {
            "descriptions": {
                "ja": "履歴データの取得タイミング",
                "en": "timing to record history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "timingType": {
                        "type": "string",
                        "enum": [
                            "onChange",
                            "interval"
                        ]
                    },
                    "intervalValue": {
                        "type": "number"
                    },
                    "intervalUnit": {
                        "type": "string",
                        "enum": [
                            "sec",
                            "min",
                            "hour",
                            "day",
                            "month",
```

```
                            "year"
                        ]
                    }
                }
            }
        },
        "first": {
            "descriptions": {
                "ja": "最初の記録時刻",
                "en": "time of the first record"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "format": "date-time"
            }
        },
        "last": {
            "descriptions": {
                "ja": "最後の記録時刻",
                "en": "time of the last record"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "format": "date-time"
            }
        },
        "total": {
            "descriptions": {
                "ja": "履歴データの総個数",
                "en": "total count of the history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "minimum": 0,
                "multipleOf": 1
            }
        }
    },
    "actions": {
        "prepareRetrieveData": {
            "descriptions": {
                "ja": "取得用データの準備を指示する",
                "en": "Prepare data to retrieve"
```

```
            },
            "input": {
                "type": "object",
                "properties": {
                    "from": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "to": {
                        "type": "string",
                        "format": "date-time"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
                    "count": {
                        "type": "number",
                        "minimum": 0,
                        "multipleOf": 1
                    },
                    "countPerPage": {
                        "type": "number",
                        "minimum": 1,
                        "multipleOf": 1
                    }
                }
            }
        },
        "retrieve": {
            "descriptions": {
                "ja": "履歴データを取得する",
                "en": "Retrieve histories data"
            },
            "input": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
                    "page": {
                        "type": "number",
                        "minimum": 1,
                        "multipleOf": 1
                    }
```

```
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "processStatus": {
                        "type": "string",
                        "enum": [
                            "succeeded",
                            "failed",
                            "inProgress"
                        ]
                    },
                    "resourceType": {
                        "type": "string",
                        "enum": [
                            "property",
                            "action",
                            "event"
                        ]
                    },
                    "resourceName": {
                        "type": "string"
                    },
                    "data": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "time": {
                                    "type": "string",
                                    "format": "date-time"
                                },
                                "value": {
                                    "type": [
                                        "string",
                                        "number",
                                        "boolean",
                                        "object",
                                        "array"
                                    ]
                                }
                            }
                        }
                    }
                }
            }
        }
```

```
    }
}
```

レスポンス：

Property	Type	Required	Description
properties	object	Yes	「GET /histories/<history id>/properties」におけるプロパティリソースに関する定義
actions	object	Yes	「POST /histories/<history id>/actions/[prepareRetrieveData, retrieve]」におけるアクションリソースに関する定義

GET /histories/<history id>/properties

history IDで指定されたhistoryの全プロパティリソース値を取得する。各プロパティには、対象となるデバイス（IDや種類）やそのリソース対象（種類や名称）、履歴データの取得タイミング、取得期間などが含まれる。履歴データの取得タイミングは不定期のタイミングで記録される"onChange"と、一定間隔で記録される"interval"のいずれかが使用され、返却される。後者の場合、その間隔の値と時間単位が共に返却される。履歴データの保存を開始した時刻を"first"、最後に保存した時刻を"last"で返却する。"last"や"total"は、時間経過や履歴データの追加保存により、更新されても良い。

■レスポンス定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
    "deviceId": <device id>,
    "deviceType": <device type>,
    "resourceType": "property | action | event",
    "resourceName": <resource name>,
    "timing": {
        "timingType": "onChange" | "interval",
        "intervalValue": <value>,
        "intervalUnit": <time unit>
    },
    "first": <first time>,
    "last": <last time>,
    "total": <total count>
}
```

■レスポンス例1（"timingType"が"onChange"の場合）

```
{
    "descriptions": {
        "ja": "エアコンabc123の動作状態",
        "en": "operation status of air-conditioner abc123"
    },
    "deviceId": "abc123",
    "deviceType": "homeAirConditioner",
    "resourceType": "property",
    "resourceName": "operationStatus",
    "timing": {
        "timingType": "onChange"
    },
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2019-04-24T22:00:00+09:00",
    "total": 1540
}
```


■レスポンス例2（"timingType"が"interval"の場合）


```
{
    "descriptions": {
        "ja": "エアコンabc123の室内温度",
        "en": "room temperature of the air-conditioner, abc123"
    },
    "deviceId": "abc123",
    "deviceType": "homeAirConditioner",
    "resourceType": "property",
    "resourceName": "roomTemperature",
    "timing": {
        "timingType": "interval",
        "intervalValue": 30,
        "intervalUnit": "min"
    },
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2019-04-24T22:00:00+09:00",
    "total": 1133
}
```


レスポンス：

Property	Type	Required	Description
descriptions	object	Yes	登録済のhistoryに関する記述
descriptions.ja	string	Yes	登録済のhistoryの内容説明（日本語）

Property	Type	Required	Description
descriptions.en	string	Yes	登録済のhistoryの内容説明（英語）
deviceId	string	Yes	履歴対象リソースを保持するデバイスのID
deviceType	string	No	上記デバイスの種類
resourceType	string	Yes	履歴対象リソースの種類。現時点でのサポートはプロパティリソース"property"のみ（将来的にはアクションリソース"action"、イベントリソース "event"についても検討予定）
resourceName	string	Yes	履歴対象リソース名。履歴対象リソースの種類に対応し、取得対象となる履歴データのリソース名
timing	object	Yes	履歴データの取得タイミング
timing.timingType	string	Yes	履歴データの取得タイミングのタイプ。"onChange", "interval"のいずれか
timing.intervalValue	number	No	時間間隔値。上記timingTypeが"interval"の場合、必須
timing.intervalUnit	string	No	時間間隔単位。"sec", "min", "hour", "day", "month", "year" のいずれか。上記timingTypeが"interval"の場合、必須
first	string	Yes	履歴データの取得開始時刻。RFC3339（ISO 8601）準拠
last	string	Yes	履歴データの取得最新時刻。RFC3339（ISO 8601）準拠。時間経過に伴い更新されても良い。
total	number	Yes	履歴データの総件数。履歴データの追加保存により更新されても良い。

GET /histories/<history id>/properties/<property resource name>

history IDにて指定されたhistoryのプロパティリソース値の取得。

■リクエスト例

```
GET /histories/00000014/properties/deviceId
```

■レスポンス定義

```
{
    <property resource name>: <property value>
}
```

■レスポンス例

```
{
    "deviceId": "abc123"
}
```

POST /histories/<history id>/actions/prepareRetrieveData

クライアントは、history IDにて指定される履歴データセットのうち、所望する履歴データの対象期間（開始時刻 "from"、終了時刻 "to"）を指定し、サーバに対して取得対象となる履歴データサブセットの確定を要求する（対象期間はオプション）。サーバは履歴データサブセットを返却できる準備が整ったら、クライアントへdata ID（"dataId"）と総件数（"count"）、1応答あたりの件数（"countPerPage"）を返却する。リクエストボディを指定しない（省略した）場合は、すべての履歴データセットを対象とする。 "from", "to"はオプションであり、指定の仕方は下記の4通りある。表中の✓は指定時、ーは未指定時を意味する。また、"from"や "to"の指定時は、いずれも同時刻にて記録された履歴データは対象に含む（"first"や"last"も同様）。取得される履歴データは、いずれの指定であっても古い順に並んだ形で返却される。なお、前述の通り、"last"は時間経過と共に更新される場合もあるため、GET /histories/<history id>/propertiesにて取得した値と異なる可能性もあることに留意すること。

ケース	from	to	対象範囲
1	✓	✓	fromから toまでの値
2	✓	ー	fromから lastまでの値
3	ー	✓	firstから toまでの値
4	ー	ー	firstから lastまでの値（履歴データセットすべて）

■リクエスト定義

```
{
    "from": <time stamp>,
    "to": <time stamp>
}
```

■レスポンス定義

```
{
    "dataId": <data id>,
    "count": <count number>,
    "countPerPage": <count per page>
}
```

■リクエスト例（検索開始時刻を指定）

```
{
    "from": "2019-04-01T08:00:00+09:00"
}
```

■レスポンス例

```
{
    "dataId": "0023",
    "count": 120,
    "countPerPage": 50
}
```

リクエスト：

Property	Type	Required	Description
from	string	No	検索開始時刻。RFC3339（ISO 8601）準拠
to	string	No	検索終了時刻。RFC3339（ISO 8601）準拠

レスポンス：

Property	Type	Required	Description
dataId	string	Yes	取得対象履歴データサブセット取得用data ID。実行時に割り当てられる。一定時間（サーバ規定）経過後に自動で削除される
count	number	Yes	取得対象履歴データサブセット内の履歴データ総件数。リクエスト時に"count"を指定した場合は、その指定数が上限となる
countPerPage	number	Yes	1応答あたりに返却可能な履歴データの件数。サーバが一度に応答できる履歴データの最大数（単位）となる。

POST /histories/<history id>/actions/retrieve

クライアントは、data IDで指定される履歴データサブセットの取得を実行する。 履歴データサブセット内の履歴データの総件数が、1応答あたりに返却可能な履歴データの件数を上回る場合は、履歴データサブセットは複数の応答（ページ）に分割される。クライアントから履歴データサブセットを取得するには、data IDに加え、ページ番号を指定する必要がある。ただし、1ページ目についてはページ番号の指定を省略することができる。 レスポンスでは、サーバでの処理状態（"processStatus"）が返却され、成功終了時（"succeeded"）には、これに加え、履歴データのリソース種（"property"、"action"、"event"のいずれか。ただし、現在、プロパティリソース"property"のみサポート）、履歴データのリソース名、履歴データ（"data"）がページ単位にて返却される。履歴データは取得時刻（"time"）と取得値（"value"）から構成される。取得値はリソース種・リソース名に応じて、様々なデータ型をとりうる。 最終ページを超えるページ番号を指定した場合は、HTTPステータスコード404(Not Found)を返す。 "processStatus"は、本操作で返却される履歴データのサーバ処理状態を示す。履歴データを含む応答が返却される場合は"succeeded"、サーバ都合で準備中の場合は"inProgress"、サーバ処理が失敗／タイムアウト（値はサーバにより規定）した場合は"failed"を、いずれもHTTPステータスコード200で返す。bulksのケースと異なり、クライアントからの中断指示機能はサポートしていないため、"aborted"は使用しない。 data ID（および対応する履歴データサブセット）は、サーバ規定による一定期間経過後に解放される。

■リクエスト定義

```
{
    "dataId": <data id>,
    "page": <page number>
}
```

■リクエスト例

```
{
    "dataId": "0023",
    "page": 2
}
```

■レスポンス定義

```
{
    "processStatus": <process status in this history session operation>,
    "resourceType": "property",
    "resourceName": <resource name>,
    "data": [
        {
            "time": <time stamp>,
            "value": <recorded resource value>
        },
        ...
```

```
    ]
}
```

■レスポンス例（成功時）

```
{
    "processStatus": "succeeded",
    "resourceType": "property",
    "resourceName": "roomTemperature",
    "data": [
        {"time": "2019-04-01T08:xx:xx+09:00", "value": 18},
        {"time": "2019-04-01T08:xy:yy+09:00", "value": 19},
        {"time": "2019-04-01T08:zz:zz+09:00", "value": 21},
        ...
    ]
}
```

■レスポンス例（サーバ処理中にて待ち状態）

```
{
    "processStatus": "inProgress"
}
```

■レスポンス例（サーバ処理失敗またはタイムアウト）

```
{
    "processStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	取得対象履歴データサブセット取得用data ID
page	number	No	履歴データサブセットの分割されたページ番号。1ページ目については省略可能

レスポンス：

Property	Type	Required	Description

Property	Type	Required	Description
processStatus	string	Yes	全体の進行状況。"inProgress"(サーバ処理中), "succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了)のいずれか。
以下は processStatus が"succeeded"の場合のみ使用。 Requiredは使用時の基準となる。			
resourceType	string	Yes	履歴データのリソース種。プロパティリソース "property"（将来的には"action"、"event"についても検討予定）
resourceName	string	Yes	履歴データのリソース名
data	array	Yes	取得対象履歴データサブセット内の履歴データを配列にて列挙。総件数が0の場合は空（"data": []）
data[].time	string	No	取得時刻。RFC3339（ISO 8601）準拠
data[].value	*1	No	取得値。データ欠損している場合は省略される（"time"は省略されない）

※：*1は、"string","number","boolean","object","array"のいずれかの型。

7.4 複数リソース対応履歴データ（resHistories）

7.3では、ある対象機器の1つの指定リソースに関する取得値を取得時刻とともに蓄積し、クライアントから検索可能な履歴データを提供するサービス種としてhistoriesを提供しているが、ここでは、対象を機器に限定せず、複数の指定リソースに関する取得値を同一タイミングにて取得時刻とともに扱える複数リソース対応の履歴データ（resource histories：サービス種名resHistories）を提供する。

複数リソース対応の履歴データは、1つ以上の取得リソース値と取得時刻の組から構成される。サーバは、履歴データ保存対象となるリソースや履歴データの記録タイミング・記録期間などを含んだ履歴データ属性（resHistory）に関する管理を自ら行う。今回のバージョンでは、クライアントから履歴データ保存対象となる機器や取得リソースをサーバへ指定・登録する手段は提供していない。クライアントに対しては、サーバが独自に規定・保存した対象や期間に基づき提供される履歴データセットについて検索手段を提供するのみとする。履歴データセットは、あるresHistory IDに基づきサーバ上で随時保存される全期間の履歴データ集合となる。

図 7-9を用いて、resHistory操作に関する一連の動作について例示する。

- クライアントはサーバに対して、①resHistory IDの一覧を取得要求可能であり、②サーバから得られたresHistory ID一覧の中から③特定のresHistory IDを指定して同IDに紐付く履歴データの定義情報（resource history description）を取得できる（④）。
- 続いて、クライアントは⑤パス上にてresHistory IDに続き"properties"を指定することで、サーバから同IDに紐付くresHistoryの全リソースプロパティ値を取得できる（⑥）。
- クライアントはサーバに対して、⑦resHistory IDを指定して（必要に応じて期間や件数を含めた）prepareRetrieveDataアクションを実行することで、サーバが提供する履歴データセットの中からクライアント所望の履歴データサブセットを返却できるよう準備開始を指示する。⑧サーバは、同準備が完了すると、このアクションに紐づくdata IDに加え、範囲確定された履歴データの総件数や1応答（ページ）あたりの件数をクライアントへ返却する（⑨）。
- 以降、⑩クライアントはdata IDとページ番号を指定し retrieveアクションを実行することで、サーバより所望の履歴データサブセットを取得する（⑪）。
- 対象となる履歴データの総件数が1応答あたりの件数よりも多い場合は、履歴データサブセットは分割されるため、⑫2ページ目以降の番号を指定することで後続の履歴データサブセットを取得でき（⑬）、⑭終了ページ番号目（（総件数）／（1応答あたりの件数）：小数点以下切り上げ）にて最終となる履歴データサブセットを取得できる。
- 履歴データサブセットの有効期間は、サーバにより自由に規定可能とするが、通常は、クライアントによる retrieveアクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。

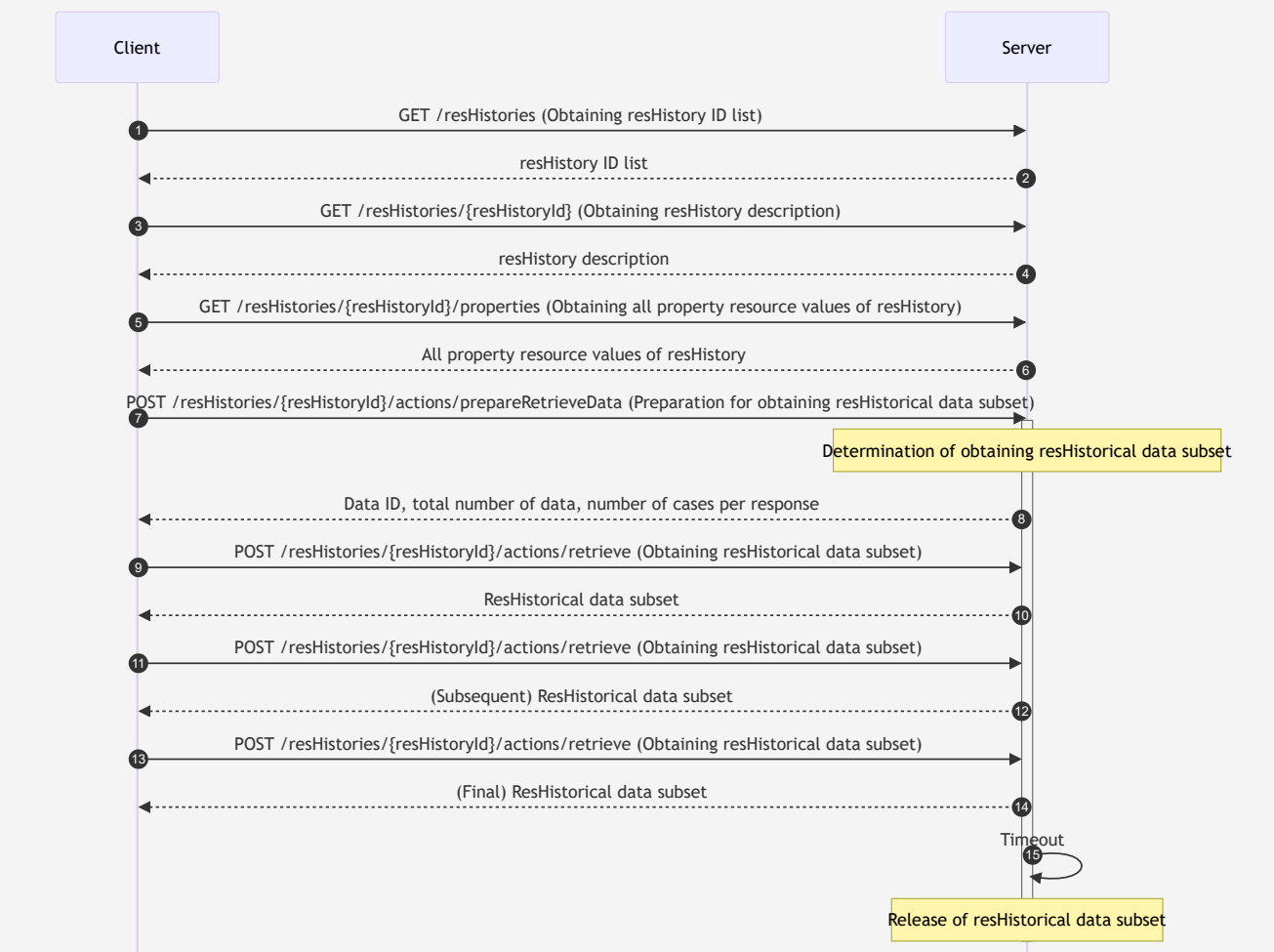


図 7-9 resource histories実行の全体概要

サーバ規定による履歴データセットと、クライアント指定による履歴データサブセットとの関係は、historiesと同様、図 7-8を用いて説明した通りとなる。

表 7-4に、resHistories関するAPI一覧を示す。以降、個別に説明する。

表 7-4　複数リソース対応の履歴データの関するAPI

http method	path	description
GET	/resHistories	resource history IDの一覧取得
GET	/resHistories/<resHistory id>	resource history description取得
GET	/resHistories/<resHistory id>/properties	resHistoryの全プロパティリソース値の取得
POST	/resHistories/<resHistory id>/actions/prepareRetrieveData	履歴データサブセット取得の準備指示
POST	/resHistories/<resHistory id>/actions/retrieve	履歴データサブセットの取得

*）なお、resHistoryのプロパティリソース値の設定は、resHistoryにて書き込み可能なプロパティリソースが存在しないため省略。

GET /resHistories

履歴IDの一覧取得。サーバ登録されている履歴IDが配列形式にて返却される。登録が無い場合は、空の配列が返却される。

■レスポンス定義

```
{
    "resHistories": [
        {
            "id": <resHistory id>,
            "descriptions": {
                "ja": <description in Japanese>,
                "en": <description in English>
            }
        },
        ...
    ]
}
```

■レスポンス例

```
{
    "resHistories": [
        {
            "id": "00000011",
            "descriptions": {
                "ja": "照明000023の動作状態履歴",
                "en": "Operation status history of light 000023"
            }
        },
        {
            "id": "00000012",
            "descriptions": {
                "ja": "蓄電池000067のAC積算放電電力量計測値[kWh]と蓄電残量
1[kWh]の30分間隔履歴",
                "en": "30-minute interval history of AC measured
cumulative discharging electric energy [kWh] and remaining stored
electricity 1 [Wh] of storage battery 000078"
            }
        }
    ]
}
```


レスポンス：

Property	Type	Required	Description
resHistories	array	Yes	それぞれの複数リソース対応履歴データに対応する id, descriptions を property とする object を列挙する。複数機器履歴データ が存在しない場合は、要素が空の array をレスポンスする。
resHistories[].id	string	No	resHistory ID
resHistories[].descriptions	object	No*1	登録済のresHistoryに関する記述
resHistories[].descriptions.ja	string	No*1	登録済のresHistoryの内容説明（日本語）
resHistories[].descriptions.en	string	No*1	登録済のresHistoryの内容説明（英語）

*1) resHistories[].id が存在する場合に必須

GET /resHistories/<resHistory id>

resHistory idで指定されたresource history descriptionを取得する。

■レスポンス例


```
{
    "properties": {
        "descriptions": {
            "descriptions": {
                "ja": "resource historyの説明",
                "en": "explanation of resource history"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "ja": {
                        "type": "string"
                    },
                    "en": {
                        "type": "string"
                    }
                }
            }
        },
        "resourceType": {
            "descriptions": {
                "ja": "リソースタイプ",
                "en": "resource type"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
                    "devices"
                ]
            }
        },
        "resourceId": {
            "descriptions": {
                "ja": "リソースID",
                "en": "resource ID"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
```

```
        "valueUnit": {
            "descriptions": {
                "ja": "値の単位の組",
                "en": "array of value units"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "array",
                "items": {
                    "type": "string",
                    "enum": [
                        "Celsius",
                        "ppm",
                        "W",
                        "kW",
                        "Wh",
                        "kWh",
                        "mA",
                        "A",
                        "Ah",
                        "V",
                        "second",
                        "minute",
                        "hour",
                        "days",
                        "degree",
                        "L",
                        "MJ",
                        "m3/h",
                        "m3",
                        "kvarh",
                        "lux",
                        "klux",
                        "r/min",
                        "%",
                        "none"
                    ]
                }
            }
        },
        "valueKind": {
            "descriptions": {
                "ja": "値の種類の組",
                "en": "array of value types"
            },
            "writable": false,
            "observable": false,
            "schema": {
```

```
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        },
        "timing": {
            "descriptions": {
                "ja": "履歴データの取得タイミング",
                "en": "timing to record history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "timingType": {
                        "type": "string",
                        "enum": [
                            "onChange",
                            "interval"
                        ]
                    },
                    "intervalValue": {
                        "type": "number"
                    },
                    "intervalUnit": {
                        "type": "string",
                        "enum": [
                            "second",
                            "minute",
                            "hour",
                            "day",
                            "month",
                            "year"
                        ]
                    }
                }
            }
        },
        "first": {
            "descriptions": {
                "ja": "最初の記録時刻",
                "en": "time of the first record"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
```

```
                "format": "date-time"
            }
        },
        "last": {
            "descriptions": {
                "ja": "最後の記録時刻",
                "en": "time of the last record"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "format": "date-time"
            }
        }
    },
    "actions": {
        "prepareRetrieveData": {
            "descriptions": {
                "ja": "取得用データの準備を指示する",
                "en": "Prepare data to retrieve"
            },
            "input": {
                "type": "object",
                "properties": {
                    "from": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "to": {
                        "type": "string",
                        "format": "date-time"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
                    "count": {
                        "type": "number",
                        "minimum": 0,
                        "multipleOf": 1
                    },
                    "countPerPage": {
                        "type": "number",
                        "minimum": 1,
```

```
                    "multipleOf": 1
                }
            }
        }
    },
    "retrieve": {
        "descriptions": {
            "ja": "履歴データを取得する",
            "en": "Retrieve histories data"
        },
        "input": {
            "type": "object",
            "properties": {
                "dataId": {
                    "type": "string"
                },
                "page": {
                    "type": "number",
                    "minimum": 1,
                    "multipleOf": 1
                }
            }
        },
        "schema": {
            "type": "object",
            "properties": {
                "processStatus": {
                    "type": "string",
                    "enum": [
                        "succeeded",
                        "failed",
                        "inProgress"
                    ]
                },
                "values": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "at": {
                                "type": "string",
                                "format": "date-time"
                            },
                            "values": {
                                "type": "object",
                                "properties": {
                                    "kind": {
                                        "type": "string"
                                    },
```

```
                                    "value": {
                                        "type": [
                                            "string",
                                            "number",
                                            "boolean",
                                            "object",
                                            "array"
                                        ]
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

レスポンス：

Property	Type	Required	Description
properties	object	Yes	「GET /resHistories/<resHistory id>/properties」におけるプロパティリソースに関する定義
actions	object	Yes	「POST /resHistories/<resHistory id>/actions」におけるプロパティリソースに関する定義

GET /resHistories/<resHistory id>/properties

resHistory IDで指定されたresHistoryの全プロパティリソース値を取得する。各プロパティには、対象となるリソース（IDや種類）やそのリソース対象（種類や名称）、履歴データの取得タイミング、取得期間などが含まれる。履歴データの取得タイミングは不定期のタイミングで記録される"onChange"と、一定間隔で記録される"interval"のいずれかが使用され、返却される。後者の場合、その間隔の値と時間単位が共に返却される。履歴データの保存を開始した時刻を"first"、最後に保存した時刻を"last"で返却する。"last"は、時間経過や履歴データの追加保存により、更新されても良い。

■レスポンス定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
```

```
    "resourceType": <resource type >,
    "resourceId": <resource id>,
    "valueUnit": [<array of value unit>],
    "valueKind":[<array of value kind>],
    "timing": {
        "timingType": "onChange" |"interval",
        "intervalValue": <value>,
        "intervalUnit": <time unit>
    },
    "first": <first time>,
    "last": <last time>}
```

■レスポンス例1（"timingType"が"onChange"の場合）

```
{
    "descriptions": {
        "ja": "エアコンabc123の動作状態",
        "en": "operation status of air-conditioner abc123"
    },
    "resourceType": "devices","resourceId": "abc123",
    "valueUnit": ["none"],
    "valueKind": ["/properties/operationStatus"],
    "timing": {
        "timingType": "onChange"
    },
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2019-04-24T22:00:00+09:00"
}
```

■レスポンス例2（"timingType"が"interval"の場合）

```
{
    "descriptions": {
        "ja": "エアコンabc123の室内温度",
        "en": "room temperature of the air-conditioner, abc123"
    },
    "resourceType": "devices",
    "resourceId": "abc123",
    "valueUnit": ["Celsius"],
    "valueKind": ["/properties/roomTemperature"],
    "timing": {
        "timingType": "interval",
        "intervalValue": 30,
        "intervalUnit": "minute"
    },
```

```
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2019-04-24T22:00:00+09:00"
}
```

■レスポンス例3（"timingType"が"interval"であり、"valueUnit"と"valueKind"が複数の場合）

```
{
    "descriptions": {
        "ja": "蓄電池000067のAC積算放電電力量計測値[kWh]と蓄電残量1[kWh]の30分間隔履歴",
        "en": "30-minute interval history of AC measured cumulative discharging electric energy [kWh] and remaining stored electricity 1 [Wh] of storage battery 000078"
    },
    "resourceType": "devices",
    "resourceId": "def456",
    "valueUnit": ["kWh", "kWh"],
    "valueKind": ["/properties/acCumulativeDischargingElectricEnergy","/properties/remainingCapacity1"],
    "timing": {
        "timingType": "interval",
        "intervalValue": 30,
        "intervalUnit": "minute"
    },
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2019-04-24T22:00:00+09:00"
}
```

レスポンス：

Property	Type	Required	Description
descriptions	object	Yes	登録済のresHistoryに関する記述
descriptions.ja	object	Yes	登録済のresHistoryの内容説明（日本語）
descriptions.en	string	Yes	登録済のresHistoryの内容説明（日本語）
resourceType	string	Yes	履歴対象リソースの種類。デバイスの場合は"devices"（*1）。
resourceId	string	Yes	履歴対象リソースを保持するリソースのID。デバイスの場合はデバイスID（*2）。

Property	Type	Required	Description
valueUnit	string	Yes	履歴値の単位の組
valueUnit[]	array[string]	Yes	履歴値の単位。上記レスポンス例では機器仕様部において規定済の単位を列挙しているが、当該履歴対象リソースの履歴値に関わる単位のみ記載する形で良い。単位が無い場合は"none"を指定。
valueKind	string	Yes	履歴値の種類の組
valueKind[]	array[string]	Yes	履歴値の種類。履歴対象リソース内にて履歴対象とする履歴値の種類を指定。履歴対象リソース毎に規定される(*3)。デバイスの場合はプロパティリソース（"/properties/<property resource name>"）を指定可能。
timing	string	Yes	履歴データの取得タイミング
timing.timingType	object	Yes	履歴データの取得タイミングのタイプ。"onChange", "interval"のいずれか
timing.intervalValue	string	Yes	時間間隔値。上記 timingType が"interval"の場合、必須
timing.intervalUnit	number	No	時間間隔単位。"second", "minute", "hour", "day", "month", "year" のいずれか。上記timingTypeが"interval"の場合、必須。対応していない単位は列挙不要。
first	string	No	履歴データの取得開始時刻。RFC3339（ISO 8601）準拠
last	string	Yes	履歴データの取得最新時刻。RFC3339（ISO 8601）準拠

*1）参考：別途定義されるDemand Response関連リソースにおけるサービス種 drResourcesを用いる場合、"drResources"。 *2）drResourcesでは、drResource IDとなる。 *3）drResourcesでは、"drCapacity", "reference"などが規定されている。

POST /resHistories/<resHistory id>/actions/prepareRetrieveData

クライアントは、resHistory IDにて指定される履歴データセットのうち、所望する履歴データの対象期間（開始時刻 "from"、終了時刻 "to"）を指定し、サーバに対して取得対象となる履歴データサブセットの確定を要求する（対象期間はオプション）。historiesにおける prepareRetrieveDataアクションと同様の処理内容となる。

POST /resHistories/<resource history id>/actions/retrieve

クライアントは、data IDで指定される履歴データサブセットの取得を実行する。

履歴データサブセット内の履歴データの総件数が、1応答あたりに返却可能な履歴データの件数を上回る場合は、履歴データサブセットは複数の応答（ページ）に分割される。クライアントから履歴データサブセットを取得するには、data IDに加え、ページ番号を指定する必要がある。ただし、1ページ目についてはページ番号の指定を省略することができる。

レスポンスでは、サーバでの処理状態（"processStatus"）が返却され、成功終了時（"succeeded"）には、これに加え、履歴データ（"values"）がページ単位にて返却される。履歴データは取得時刻（"at"）と取得値（"values"）の組の配列から構成される。取得値はリソース種・リソース名に応じて、様々なデータ型をとりうる。 最終ページを超えるページ番号を指定した場合は、HTTPステータスコード404(Not Found)を返す。

"processStatus"は、本操作で返却される履歴データのサーバ処理状態を示す。履歴データを含む応答が返却される場合は"succeeded"、サーバ都合で準備中の場合は"inProgress"、サーバ処理が失敗／タイムアウト（値はサーバにより規定）した場合は"failed"を、いずれもHTTPステータスコード200で返す。bulksのケースと異なり、クライアントからの中断指示機能はサポートしていないため、"aborted"は使用しない。

data ID（および対応する履歴データサブセット）は、サーバ規定による一定期間経過後に解放される。

■リクエスト定義

```
{
    "dataId": <data id>,
    "page": <page number>
}
```

■リクエスト例

```
{
    "dataId": "0023",
    "page": 2
}
```

■レスポンス定義

```
{
    "processStatus": <process status in this history session operation>,
    "values": [
    {
            "at": <time stamp>,
            "values": [<recorded value1>, <recorded value2>, …]
        },
```

```
        ...
    ]
}
```

■レスポンス例（成功時）

```
{
    "processStatus": "succeeded",
    "values": [
        {"at": "2019-04-01T08:xx:xx+09:00", "values": [18, 1]},
        {"at": "2019-04-01T08:xy:yy+09:00", "values": [19, 2]},
        {"at": "2019-04-01T08:zz:zz+09:00", "values": [21, 3]},
        ...
    ]
}
```

■レスポンス例（サーバ処理中にて待ち状態）

```
{
    "processStatus": "inProgress"
}
```

■レスポンス例（サーバ処理失敗またはタイムアウト）

```
{
    "processStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	取得対象履歴データサブセット取得用data ID
page	number	No	履歴データサブセットの分割されたページ番号。1ページ目については省略可能

レスポンス：

Property	Type	Required	Description

Property	Type	Required	Description
processStatus	string	Yes	全体の進行状況。"inProgress"(サーバ処理中), "succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了)のいずれか。
以下は processStatus が"succeeded"の場合のみ使用。 Requiredは使用時の基準となる。			
values	array	Yes	取得対象履歴データサブセット内の履歴データを配列にて列挙。総件数が0の場合は空（"values": []）
values[].at	string	No	取得時刻。RFC3339（ISO 8601）準拠
values[].values	*1	No	取得値。データ欠損している場合は省略される（"at"は省略されない）

※：*1は、"string","number","boolean","object","array"のいずれかの型。

7.5 複数機器履歴データ（groupHistories）

7.3 では、ある対象機器の 1 つの指定リソースに関する取得値を取得時刻とともに蓄積し、クライアントから検索可能な履歴データを提供するサービス種として histories を提供している。一方、本節においては、複数の機器の 1 つの指定リソースに関する取得値を取得時刻とともに蓄積し、クライアントから検索可能な複数機器履歴データを提供するサービス種として groupHistories を提供している。なお、複数の機器を指定する方法として、対象機器を識別する deviceId を指定する方法と機器のグルーピングを識別する groupId を用いる方法を規定する。なお、機器のグルーピングを識別する groupId は、「7.2　機器のグルーピング（groups)」にて「識別子（group ID)」として定義されている。

本節で提供する複数機器履歴データは、対象機器、取得リソース値および取得時刻の組から構成される。サーバは、複数機器履歴データ保存対象となる機器や複数機器履歴データの記録タイミング・記録期間などを含んだ複数機器履歴データ属性（groupHistory）に関する管理を自ら行う。複数機器履歴データセットは、ある group history ID に基づき、サーバ上で随時保存される複数機器履歴データ集合となる。

クライアントに対しては、サーバが規定・保存した対象や期間に基づき提供される履歴データセットについて検索手段を提供する。なお、履歴データセットの対象となる機器の情報などについては、Web API を用いてサーバ・クライアント間で共有せずに、事前に契約などの別手段で情報共有するユースケースも想定している。なお、履歴対象機器の種類(deviceType)、履歴対象機器の ID(deviceId)及び履歴対象グループの ID(groupId)は groupHistory の property には含めない。これらを取得する手段として action: getDeviceType, getDeviceId, getGroupId をオプション仕様として定義する。

本節で提供する複数機器履歴データ（複数機器一括取得）について、クライアント視点においては以下のニーズへ対応する。

- 複数の機器の履歴データを一括で取得したい
- 大容量の履歴データを取得したい

また、サーバ視点においては以下のニーズへ対応する。

- 従来の historyId の管理コスト・生成コストを下げたい
- 機器の構成などが変わっても、管理する履歴データのプロパティリソース値を変更する必要が無いようにしたい
- 履歴データを相対的な期間設定することで、保持する応答データ容量を抑制したい
- 履歴データの通信量についても抑制したい
- 大容量の履歴データへの対応として、履歴データの量によって提供方法を柔軟にしたい

複数機器履歴データ（groupHistories）の主な特長を以下に示す。

- 新たに規定する group history ID は、プロパティリソース値の管理コストを削減するために、プロパティリソースに deviceId, deviceType を含めない。機器の買い替え、新規購入などにより、システム構成が変更することは容易に起こりえるため、これらの変更時において、プロパティリソース値の更新を不要とする。
- groupHistory ID は複数の機器の特定のリソースの履歴データに対応する。リソースの property 名が同一であれば異なる機器種の履歴データも扱うことができる。(例１：共通項目の faultStatus, 例２：エアコンや蓄電池の operationMode)。また、履歴データのデータ構造は、time stamp をキー とする ″time″ 型と device ID をキーとする ″device″ 型を定義する。
- 複数の機器の履歴データを一括で取得することから、複数機器履歴データの通信において、様々なデータ量が想定される。またサーバもしくはクライアントにとっても処理したいデータ形式も異なることが想定される。そのため、複数機器履歴データの取り扱いには、「同期型」、「非同期型」、「ファイル型」の複数の方式を規定する。なお、「同期型」、「非同期型」、「ファイル型」はサーバ、クライアントともに、任意の 1 つないし複数の方式をサポートすれば良く、必ずしもすべての方式をサポートする必要は無い。
- サーバでの保存データ量の抑制を可能にするために、サーバが保存する履歴データの期間に関して、現在時点を基準として動的に保存期間が決定される″duration″という仕様（例：最新の３ヶ月間）を追加する。

図 7-10 ～図 7-13 を用いて、groupHistory 操作に関する一連の動作について例示する。 図 7-10 は、「同期型」、「非同期型」、「ファイル型」の共通の動作であり、共通部の後の個別の動作を図 7-11、図 7-12、図 7-13 に示す。

【共通】

- クライアントはサーバに対して、①group history ID の一覧を取得要求可能であり、② サーバから得られた group history ID 一覧の中から ③ 特定の group history ID を指定して同 ID に紐付く複数機器履歴データの定義情報（groupHistory description）を取得できる（④）。
- 続いて、クライアントは ⑤ パス上にて group history ID に続き″properties″を指定することで、サーバから同 ID に紐付く groupHistory の全リソースプロパティ値を取得できる（⑥）。

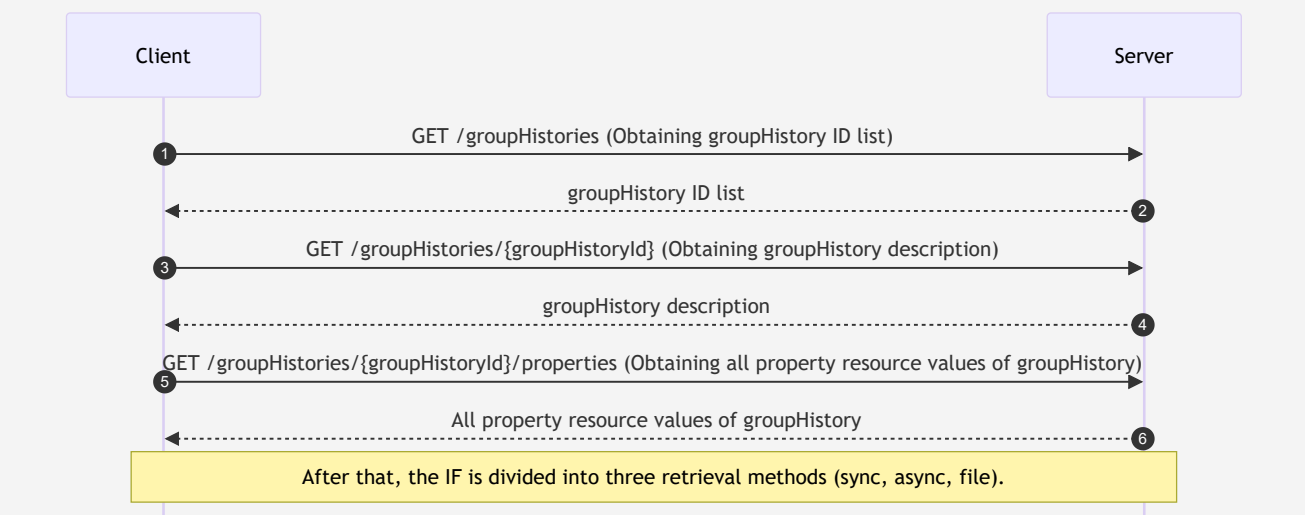


図 7-10 groupHistories 共通部の全体概要

以下、「同期型」、「非同期型」、「ファイル型」の動作について、順に説明する。まずは、同期型（sync）のシーケンスについて説明する。

【同期型 (sync)】

- クライアントはサーバに対して、⑦group history ID を指定して（必要に応じて期間やデータ型を含めた） prepareRetrieveDataBySync アクションを実行することで、サーバが提供する複数機器履歴データセットの中 からクライアント所望の複数機器履歴データサブセットを返却できるよう準備開始を指示する。⑧ サーバは、同準備が完了すると、このアクションに紐づく data ID に加え、範囲確定された複数機器履歴データの総件数や 1 応答（ページ）あたりの件数をクライアントへ返却する。
- ⑨ クライアントは data ID とページ番号を指定し retrieveBySync アクションを実行すること で、サーバより所望の複数機器履歴データサブセットを取得する（⑩）。
- 対象となる複数機器履歴データの総件数が 1 応答あたりの件数よりも多い場合は、複数機器履歴データサブセットは分割されるため、⑨、⑩ の処理を繰り返すことで、2 ページ目以降の番号を指定することで後続の複数機器履歴データサブセット を取得でき、終了ページ番号目（（総件数）／（1 応答あたりの件数）：小数点以下切り 上げ）にて最終となる複数機器履歴データサブセットを取得できる。
- 複数機器履歴データサブセットの有効期間は、サーバにより自由に規定可能とするが、通常は、クライアントによる retrieveBySync アクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。

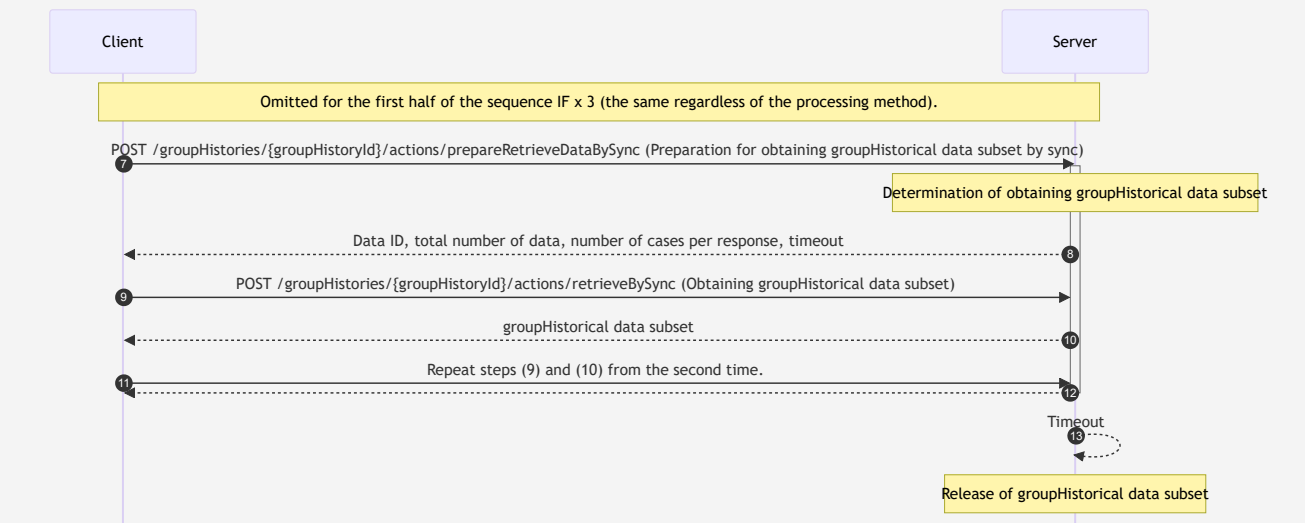


図 7-11 groupHistories「同期型（sync）」の全体概要

次に非同期型（async）のシーケンスについて説明する。

【非同期型 (async)】

- クライアントはサーバに対して、⑦group history ID を指定して（必要に応じて期間やデータ型を含めた） `prepareRetrieveDataByAsync` アクションを実行することで、サーバが提供する複数機器履歴データセットの中からクライアント所望の複数機器履歴データサブセットを返却できるよう準備開始を指示する。⑧ サーバは、複数機器履歴データサブセット作成と並行して、このアクションに紐づく data ID に加え、複数機器履歴データサブセット作成完了推定時刻、タイムアウト時刻（複数機器履歴データサブセット作成後、複数機器履歴データサブセットを取得可能な時刻）をクライアントへ返却する。
- 「範囲確定された複数機器履歴データの総件数」をレスポンスするために時間がかかる場合を想定し、`action:inquireByAsync` を追加する。
- ⑨ クライアントは、複数機器履歴データサブセット作成完了推定時刻を経過後もしくは一定時間経過後、`dataId` を指定し `inquireByAsync` アクションを実行することで、複数機器履歴データサブセット作成状況を取得する。⑩ サーバは、複数機器履歴データサブセット作成が終了していた場合、範囲確定された複数機器履歴データの総件数、1 応答（ページ）あたりの件数、タイムアウト時刻をクライアントへ返却する。
- ⑪ クライアントは data ID とページ番号を指定し `retrieveByAsync` アクションを実行すること で、サーバより所望の複数機器履歴データサブセットを取得する（⑫）。
- 対象となる複数機器履歴データの総件数が 1 応答あたりの件数よりも多い場合は、複数機器履歴データサブセットは分割されるため、⑪、⑫ の処理を繰り返すことで、2 ページ目以降の番号を指定することで後続の複数機器履歴データサブセット を取得でき、終了ページ番号目（（総件数）／（1 応答あたりの件数）：小数点以下切り 上げ）にて最終となる複数機器履歴データサブセットを取得できる。
- 複数機器履歴データサブセットの有効期間は、サーバにより自由に規定可能とするが、通常は、クライアントによる `retrieveByAsync` アクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。

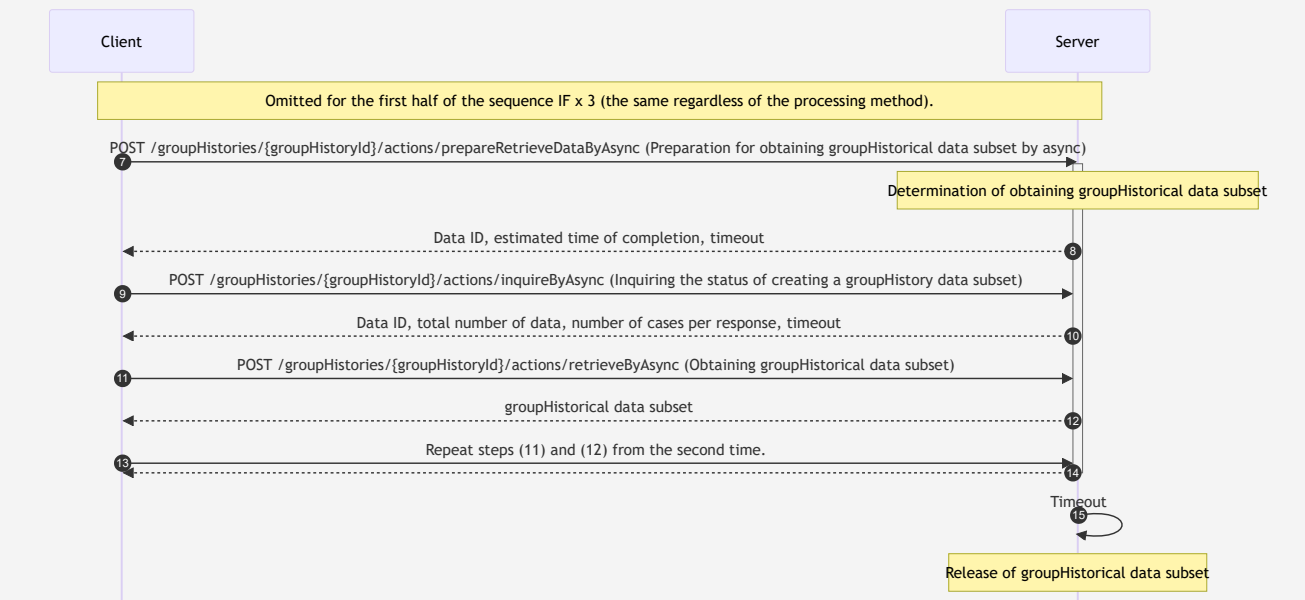


図 7-12 groupHistories「非同期型（async）」の全体概要

最後にファイル型（file）のシーケンスについて説明する。

【ファイル型（file)】

- クライアントはサーバに対して、⑦group history ID を指定して（必要に応じて期間やデータ型を含めた） `prepareRetrieveDataByFile` アクションを実行することで、サーバが提供する複数機器履歴データセットの中 からクライアント所望の複数機器履歴データサブセットを返却できるよう準備開始を指示する。⑧ サーバは、複数機器履歴データサブセット作成と並行して、このアクションに紐づく data ID に加え、複数機器履歴データサブセット作成完了推定時刻、タイムアウト時刻（複数機器履歴データサブセット作成後、複数機器履歴データサブセットを取得可能な時刻）をクライアントへ返却する。
- ⑨ クライアントは data ID を指定し `retrieveByFile` アクションを実行することで、サーバより所望の複数機器履歴データサブセットを取得するための URL を取得する（⑩）。
- ⑪ クライアントは上記 URL にアクセスすることで、複数機器履歴データを取得する。
- 複数機器履歴データサブセットの有効期間は、サーバにより自由に規定可能とするが、通常は、クライアントによる `retrieveByFile` アクションを用いた結果取得が余裕をもって実施できる十分な期間が設定されることが望ましい。

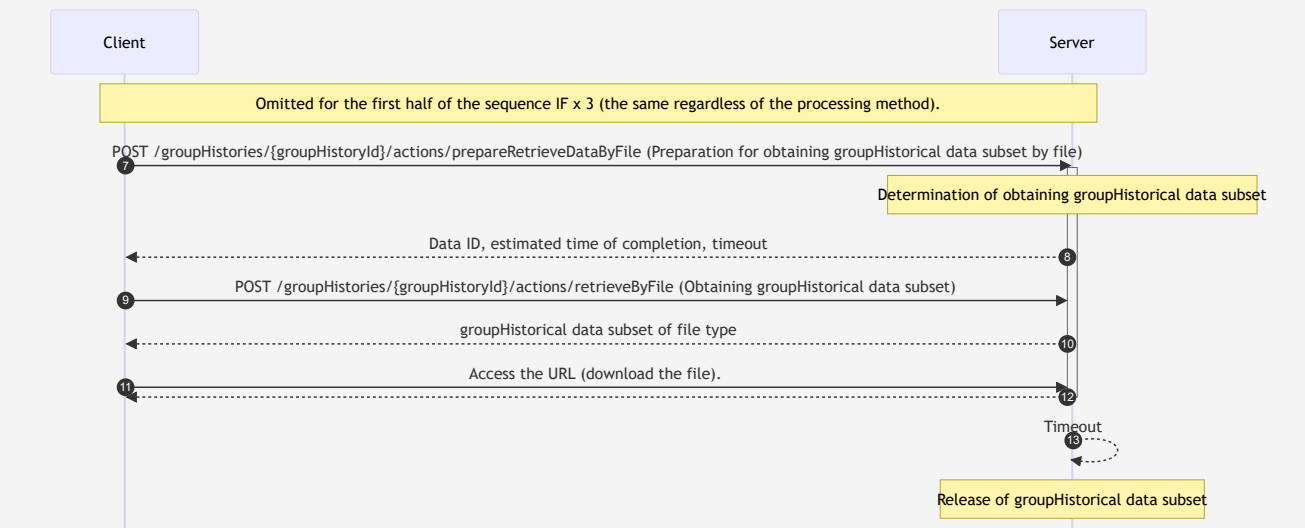


図 7-13 groupHistories「ファイル型（file）」の全体概要

複数機器履歴データサブセットの期間設定時に「duration」を用いる場合のサーバ規定による複数機器履歴データセットと、クライアント指定による複数機器履歴データサブセットとの関係について説明する。「duration」は、サーバが保存する複数機器履歴データの期間に関して、現在時点を起点とした保存開始期間を決定する。なお、「duration」の単位は、day, month, year とする。単位が「day」の場合、指定された日の 0 時 0 分～現在時点までのデータを複数機器履歴データとして扱う。単位が「month」の場合、指定された月の 1 日 0 時 0 分～現在時点までのデータを複数機器履歴データとして扱う。単位が「year」の場合、指定された年の 1 月の 1 日 0 時 0 分～現在時点までのデータを複数機器履歴データとして扱う。 また、複数機器履歴データサブセットの期間設定時に「first」and/or「last」を用いる場合については、図 7-8 にて定義している。

例えば、duration = 3 years(value=3, unit=year)が設定されている group history ID に対して、2023 年 8 月 23 日時点では、2020 年 1 月 1 日 0 時 0 分～現在までの複数機器履歴データをサーバが保持する。2023 年 12 月 31 日までは、 2020 年 1 月 1 日 0 時 0 分～現在までの複数機器履歴データであり、2024 年 1 月 1 日なると、2021 年 1 月 1 日 0 時 0 分～現在までの複数機器履歴データとなる。 また、duration = 2 months(value=2, unit=month)が設定されている group history ID に対して、2023 年 8 月 23 日時点では、6 月 1 日 0 時 0 分～現在までの複数機器履歴データをサーバが保持する。8 月 31 日までは、 6 月 1 日 0 時 0 分～現在までの複数機器履歴データであり、9 月 1 日になると、7 月 1 日 0 時 0 分～現在までの複数機器履歴データとなる。 同様に、duration = 4 days(value=4, unit=day)が設定されている group history ID に対して、2023 年 8 月 23 日時点では、8 月 19 日 0 時 0 分～現在までの複数機器履歴データをサーバが保持する。8 月 24 日になると、8 月 20 日 0 時 0 分～現在までの複数機器履歴データとなる。

表 7-5 に、groupHistories に関する API 一覧を示す。以降、個別に説明する。

表 7-5　複数機器履歴データAPI一覧

Method	Endpoint	Description
GET	/groupHistories	groupHistory ID の一覧取得
GET	/groupHistories/<group history id>	groupHistory description 取得

Method	Endpoint	Description
GET	/groupHistories/<group history id>/properties	groupHistory の全プロパティリソース取得
POST	/groupHistories/<group history id>/actions/prepareRetrieveDataBySync	複数機器履歴データサブセット取得の準備（sync）
POST	/groupHistories/<group history id>/actions/retrieveBySync	複数機器履歴データサブセット取得（sync）
POST	/groupHistories/<group history id>/actions/prepareRetrieveDataByAsync	複数機器履歴データサブセット取得の準備（async）
POST	/groupHistories/<group history id>/actions/inquireByAsync	複数機器履歴データサブセット作成状況取得（async）
POST	/groupHistories/<group history id>/actions/retrieveByAsync	複数機器履歴データサブセット取得（async）
POST	/groupHistories/<group history id>/actions/prepareRetrieveDataByFile	複数機器履歴データサブセット取得の準備（file）
POST	/groupHistories/<group history id>/actions/retrieveByFile	複数機器履歴データサブセット取得（file）
POST	/groupHistories/<group history id>/actions/abortPrepareRetrieveDataByAsync	prepareRetrieveDataByAsync実行中断
POST	/groupHistories/<group history id>/actions/abortPrepareRetrieveDataByFile	prepareRetrieveDataByFile実行中断
POST	/groupHistories/<group history id>/actions/getDeviceType	device typeの一覧取得
POST	/groupHistories/<group history id>/actions/getDeviceId	機器一覧取得
POST	/groupHistories/<group history id>/actions/getGroupId	group IDの一覧取得

なお、groupHistory のプロパティリソース値の設定は、groupHistory にて書き込み可能なプロパティリソースが存在しないため省略。

GET /groupHistories

groupHistory ID の一覧取得。サーバ登録されている groupHistory ID を含むオブジェクトが配列形式にて返却される。登録が無い場合は、空の配列が返却される。

■ レスポンス定義

```
{
    "groupHistories": [
```

```
        {
            "id": <groupHistoryId>,
            "descriptions": {
                "ja": <description in Japanese>,
                "en": <description in English>
            }
        },
        ...
    ]
}
```

■ レスポンス例

```
{
  "groupHistories": [
    {
      "id": "00000011",
      "descriptions": {
        "ja": "複数照明の動作状態履歴",
        "en": "Operation status history of lights"
      }
    },
    {
      "id": "00000012",
      "descriptions": {
        "ja": "複数太陽光発電（PV）の積算発電電力量計測値（kWh）",
        "en": "Cumulative amount of electric energy generated of
photovoltaics ([kWh], 30min interval) "
      }
    }
  ]
}
```

レスポンス：

Property	Type	Required	Description
groupHistories	array	Yes	それぞれの複数機器履歴データに対応する id, descriptions を property とする object を列挙する。複数機器履歴データ が存在しない場合は、要素が空の array をレスポンスする。
groupHistories[].id	string	No	groupHistory ID

Property	Type	Required	Description
groupHistories[].descriptions	object	No*1	登録済の groupHistory に関する記述
groupHistories[].descriptions.ja	string	No*1	登録済の groupHistory の内容説明（日本語）
groupHistories[].descriptions.en	string	No*1	登録済の groupHistory の内容説明（英語）

*1) groupHistories[].id が存在する場合に必須

GET /groupHistories/<group history id>

group history ID で指定された groupHistory description を取得する。

■ レスポンス例

```
{
    "properties": {
        "descriptions": {
            "descriptions": {
                "ja": "group historyの説明",
                "en": "explanation of group history"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "ja": {
                        "type": "string"
                    },
                    "en": {
                        "type": "string"
                    }
                }
            }
        },
        "resourceType": {
            "descriptions": {
                "ja": "リソースタイプ",
                "en": "resource type"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
```

```
                    "property",
                    "action",
                    "event"
                ]
            }
        },
        "resourceName": {
            "descriptions": {
                "ja": "リソース名",
                "en": "resource name"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string"
            }
        },
        "timing": {
            "descriptions": {
                "ja": "複数機器履歴データの取得タイミング",
                "en": "timing to record group history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "timingType": {
                        "type": "string",
                        "enum": [
                            "onChange",
                            "interval"
                        ]
                    },
                    "intervalValue": {
                        "type": "number"
                    },
                    "intervalUnit": {
                        "type": "string",
                        "enum": [
                            "sec",
                            "min",
                            "hour",
                            "day",
                            "month",
                            "year"
                        ]
                    }
                }
```

```
            }
        },
        "period": {
            "descriptions": {
                "ja": "複数機器履歴データの取り扱い期間",
                "en": "period of time to keep group history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "object",
                "properties": {
                    "first": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "last": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "duration": {
                        "type": "object",
                        "properties": {
                            "value": {
                                "type": "number",
                                "multipleOf": 1
                            },
                            "unit": {
                                "type": "string",
                                "enum": [
                                    "day",
                                    "month",
                                    "year"
                                ]
                            }
                        }
                    }
                }
            }
        },
        "maxRequestDuration": {
            "descriptions": {
                "ja": "複数機器履歴データの取得可能な期間",
                "en": "maximum duration for which group history data can
be retrieved"
            },
            "writable": false,
            "observable": false,
            "schema": {
```

```
                "type": "object",
                "properties": {
                    "value": {
                        "type": "number",
                        "multipleOf": 1
                    },
                    "unit": {
                        "type": "string",
                        "enum": [
                            "day",
                            "month",
                            "year"
                        ]
                    }
                }
            }
        },
        "maxRequestNumberOfDevices": {
            "descriptions": {
                "ja": "複数機器履歴データ取得可能な最大要求台数",
                "en": "maximum number of devices requested"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "number",
                "multipleOf": 1
            }
        },
        "retrieveMethod": {
            "descriptions": {
                "ja": "データ取得方法",
                "en": "method to retrieve group history data"
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "string",
                "enum": [
                    "sync",
                    "async",
                    "file"
                ]
            }
        },
        "dataStructure": {
            "descriptions": {
                "ja": "複数機器履歴データの型",
                "en": "type of group history data"
```

```
            },
            "writable": false,
            "observable": false,
            "schema": {
                "type": "array",
                "items": {
                    "type": "string",
                    "enum": [
                        "device",
                        "time"
                    ]
                }
            }
        }
    },
    "actions": {
        "prepareRetrieveDataBySync": {
            "descriptions": {
                "ja": "取得用データの準備を指示する(Sync)",
                "en": "Prepare data to retrieve by sync"
            },
            "input": {
                "type": "object",
                "properties": {
                    "groupId": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "deviceId": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "from": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "to": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "dataStructure": {
                        "type": "string",
                        "enum": [
                            "device",
                            "time"
```

```
                    ]
                }
            }
        },
        "schema": {
            "type": "object",
            "properties": {
                "dataId": {
                    "type": "string"
                },
                "count": {
                    "type": "number",
                    "minimum": 0,
                    "multipleOf": 1
                },
                "countPerPage": {
                    "type": "number",
                    "minimum": 1,
                    "multipleOf": 1
                },
                "expirationTime": {
                    "type": "string",
                    "format": "date-time"
                }
            }
        }
    },
    "retrieveBySync": {
        "descriptions": {
            "ja": "履歴データを取得する(Sync)",
            "en": "Retrieve histories data by sync"
        },
        "input": {
            "type": "object",
            "properties": {
                "dataId": {
                    "type": "string"
                },
                "page": {
                    "type": "number",
                    "minimum": 1,
                    "multipleOf": 1
                }
            }
        },
        "schema": {
            "type": "object",
            "properties": {
                "processStatus": {
```

```
                        "type": "string",
                        "enum": [
                            "succeeded",
                            "failed",
                            "aborted"
                        ]
                    },
                    "resourceType": {
                        "type": "string",
                        "enum": [
                            "property",
                            "action",
                            "event"
                        ]
                    },
                    "resourceName": {
                        "type": "string"
                    },
                    "expirationTime": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "data": {
                        "oneOf": [
                            {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "properties": {
                                        "deviceId": {
                                            "type": "string"
                                        },
                                        "deviceData": {
                                            "type": "array",
                                            "items": {
                                                "type": "object",
                                                "properties": {
                                                    "time": {
                                                        "type": "string",
                                                        "format": "date-
time"
                                                    },
                                                    "value": {
                                                        "type": [
                                                            "string",
                                                            "number",
                                                            "boolean",
                                                            "object",
                                                            "array"
```

```
                                                ]
                                            }
                                        }
                                    }
                                }
                            }
                        },
                        {
                            "type": "array",
                            "items": {
                                "type": "object",
                                "properties": {
                                    "time": {
                                        "type": "string",
                                        "format": "date-time"
                                    },
                                    "deviceId": {
                                        "type": "string"
                                    },
                                    "value": {
                                        "type": [
                                            "string",
                                            "number",
                                            "boolean",
                                            "object",
                                            "array"
                                        ]
                                    }
                                }
                            }
                        }
                    }
                ]
            }
        }
    },
    "prepareRetrieveDataByASync": {
        "descriptions": {
            "ja": "取得用データの準備を指示する(Async)",
            "en": "Prepare data to retrieve by async"
        },
        "input": {
            "type": "object",
            "properties": {
                "groupId": {
                    "type": "array",
                    "items": {
                        "type": "string"
```

```
                        }
                    },
                    "deviceId": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "from": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "to": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "dataStructure": {
                        "type": "string",
                        "enum": [
                            "device",
                            "time"
                        ]
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
                    "estimatedTimeOfCompletion": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "expirationTime": {
                        "type": "string",
                        "format": "date-time"
                    }
                }
            }
        },
        "inquireByAsync": {
            "descriptions": {
                "ja": "履歴データサブセットの取得準備状況を確認する(Async)",
                "en": "Check readiness to retrieve historical data subsets
by async"
            },
            "input": {
```

```
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "processStatus": {
                        "type": "string",
                        "enum": [
                            "succeeded",
                            "failed",
                            "inProgress",
                            "aborted"
                        ]
                    },
                    "count": {
                        "type": "number",
                        "multipleOf": 1
                    },
                    "countPerPage": {
                        "type": "number",
                        "multipleOf": 1
                    },
                    "estimatedTimeOfCompletion": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "expirationTime": {
                        "type": "string",
                        "format": "date-time"
                    }
                }
            }
        },
        "retrieveByASync": {
            "descriptions": {
                "ja": "履歴データを取得する(Async)",
                "en": "Retrieve histories data by async"
            },
            "input": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
```

```
                "page": {
                    "type": "number",
                    "minimum": 1,
                    "multipleOf": 1
                }
            }
        },
        "schema": {
            "type": "object",
            "properties": {
                "processStatus": {
                    "type": "string",
                    "enum": [
                        "succeeded",
                        "failed",
                        "inProgress",
                        "aborted"
                    ]
                },
                "resourceType": {
                    "type": "string",
                    "enum": [
                        "property",
                        "action",
                        "event"
                    ]
                },
                "resourceName": {
                    "type": "string"
                },
                "expirationTime": {
                    "type": "string",
                    "format": "date-time"
                },
                "data": {
                    "oneOf": [
                        {
                            "type": "array",
                            "items": {
                                "type": "object",
                                "properties": {
                                    "deviceId": {
                                        "type": "string"
                                    },
                                    "deviceData": {
                                        "type": "array",
                                        "items": {
                                            "properties": {
                                                "time": {
```

```
                                                "type": "string",
                                                "format": "date-
time"
                                            },
                                            "value": {
                                                "type": [
                                                    "string",
                                                    "number",
                                                    "boolean",
                                                    "object",
                                                    "array"
                                                ]
                                            }
                                        }
                                    }
                                }
                            }
                        },
                        {
                            "type": "array",
                            "items": {
                                "type": "object",
                                "properties": {
                                    "time": {
                                        "type": "string",
                                        "format": "date-time"
                                    },
                                    "deviceId": {
                                        "type": "string"
                                    },
                                    "value": {
                                        "type": [
                                            "string",
                                            "number",
                                            "boolean",
                                            "object",
                                            "array"
                                        ]
                                    }
                                }
                            }
                        }
                    ]
                }
            }
        }
    },
    "prepareRetrieveDataByFile": {
```

```
            "descriptions": {
                "ja": "取得用データの準備を指示する(File)",
                "en": "Prepare data to retrieve by file"
            },
            "input": {
                "type": "object",
                "properties": {
                    "groupId": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "deviceId": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "from": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "to": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "dataStructure": {
                        "type": "string",
                        "enum": [
                            "device",
                            "time"
                        ]
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    },
                    "estimatedTimeOfCompletion": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "expirationTime": {
                        "type": "string",
                        "format": "date-time"
```

```
                    }
                }
            }
        },
        "retrieveByFile": {
            "descriptions": {
                "ja": "履歴データを取得する(File)",
                "en": "Retrieve histories data by file"
            },
            "input": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "processStatus": {
                        "type": "string",
                        "enum": [
                            "succeeded",
                            "failed",
                            "inProgress",
                            "aborted"
                        ]
                    },
                    "resourceType": {
                        "type": "string",
                        "enum": [
                            "property",
                            "action",
                            "event"
                        ]
                    },
                    "resourceName": {
                        "type": "string"
                    },
                    "url": {
                        "type": "string"
                    },
                    "estimatedTimeOfCompletion": {
                        "type": "string",
                        "format": "date-time"
                    },
                    "expirationTime": {
                        "type": "string",
```

```
                        "format": "date-time"
                    }
                }
            }
        },
        "abortPrepareRetrieveDataByAsync": {
            "descriptions": {
                "ja": "prepareRetrieveDataByAsync実行の中断(Async)",
                "en": "Abort a prepareRetrieveDataByAsync execution by
async"
            },
            "input": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
            }
        },
        "abortPrepareRetrieveDataByFile": {
            "descriptions": {
                "ja": "prepareRetrieveDataByFile実行の中断(File)",
                "en": "Abort a prepareRetrieveDataByFile execution by
file"
            },
            "input": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
            },
            "schema": {
                "type": "object",
                "properties": {
                    "dataId": {
                        "type": "string"
                    }
                }
```

```
            }

        },
        "getDeviceType": {
            "descriptions": {
                "ja": "groupHistory IDに対応するdeviceType一覧の取得",
                "en": "Get a list of group Types for a groupHistory ID"
            },
            "schema": {
                "type": "object",
                "properties": {
                    "deviceType": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    }
                }
            }
        },
        "getDeviceId": {
            "descriptions": {
                "ja": "groupHistory IDに対応する機器一覧の取得",
                "en": "Get a list of devices for a groupHistory ID"
            },
            "schema": {
                "type": "object",
                "properties": {
                    "devices": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "id": {
                                    "type": "string"
                                },
                                "deviceType": {
                                    "type": "string"
                                },
                                "protocol": {
                                    "type": "object",
                                    "properties": {
                                        "type": {
                                            "type": "string"
                                        },
                                        "version": {
                                            "type": "string"
                                        }
                                    }
```

```
                            },
                            "manufacturer": {
                                "type": "object",
                                "properties": {
                                    "code": {
                                        "type": "string"
                                    },
                                    "descriptions": {
                                        "type": "object",
                                        "properties": {
                                            "ja": {
                                                "type": "string"
                                            },
                                            "en": {
                                                "type": "string"
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        },
        "getGroupId": {
            "descriptions": {
                "ja": "groupHistory IDに対応するgroup ID一覧の取得",
                "en": "Get a list of group IDs for a groupHistory ID"
            },
            "schema": {
                "type": "object",
                "properties": {
                    "groups": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "id": {
                                    "type": "string"
                                },
                                "descriptions": {
                                    "type": "object",
                                    "properties": {
                                        "ja": {
                                            "type": "string"
                                        },
                                        "en": {
```

```
                                        "type": "string"
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

レスポンス：

Property	Type	Required	Description
properties	object	Yes	「GET /groupHistories/<group history id>/properties」におけるプロパティリソースに関する定義
actions	object	Yes	「POST /groupHistories/<group history id>/actions/[prepareRetrieveDataBySync, prepareRetrieveDataByAsync, prepareRetrieveDataByFile, retrieveBySync, retrieveByAsync, retrieveByFile, inquireByAsync, abortPrepareRetrieveDataByAsync, abortPrepareRetrieveDataByFile, getDeviceType, getDeviceId, getGroupId]」におけるアクションリソースに関する定義

GET /groupHistories/<group history id>/properties

group history ID で指定された groupHistory の全プロパティリソース値を取得する。各プロパティには、対象となるリソース対象（種類や名称）、複数機器履歴データの取得タイミング、取得期間などが含まれる。

複数機器履歴データの取得タイミングは不定期のタイミングで記録される"onChange"と、一定間隔で記録される"interval"のいずれかが使用され、返却される。後者の場合、その間隔の値と時間単位が共に返却される。複数機器履歴データの保存期間を示す"duration"、もしくは保存を開始した時刻を"first"、最後に保存した時刻を"last"で返却する。"last"や"total"は、時間経過や複数機器履歴データの追加保存により、更新しても良い。

また、複数機器履歴データの取得可能な期間情報である"maxRequestDuration"、複数機器履歴データ取得可能な最大要求台数である"maxRequestNumberOfDevices"、データ取得方法である"retrieveMethod"、そして複数機器履歴データの型を示す"dataStructure"が返却される。

■ レスポンス定義

```
{
    "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
    },
    "resourceType": "property",
    "resourceName": <resource name>,
    "timing": {
        "timingType": "onChange" | "interval",
        "intervalValue": <value of interval>,
        "intervalUnit": "sec" | "min" | "hour" | "day" | "month" | "year"
    },
    "period": {
        "first": <time stamp of first-data>,
        "last": <time stamp of last-data>,
        "duration": {
            "value": <value of duration>,
            "unit": "day" | "month" | "year"
        }
    },
    "maxRequestDuration": {
        "value": <value of maxRequestDuration>,
        "unit": "day" | "month" | "year"
    },
    "maxRequestNumberOfDevices": <value of maxRequestNumberOfDevices>,
    "retrieveMethod": "sync" | "async" | "file",
    "dataStructure": ["device", "time"]
}
```

■ レスポンス例 1（"timingType"が"onChange"かつ"period"が"duration"の場合）

```
{
  "descriptions": {
    "ja": "エアコンの動作状態",
    "en": "operation status of the air-conditioner"
  },
  "resourceType": "property",
  "resourceName": "operationStatus",
  "timing": {
    "timingType": "onChange"
  },
  "period": {
    "duration": {
      "value": 6,
      "unit": "month"
```

```
    }
  },
  "maxRequestDuration": {
    "value": 30,
    "unit": "day"
  },
  "maxRequestNumberOfDevices": 10,
  "retrieveMethod": "async",
  "dataStructure": ["device"]
}
```

■ レスポンス例 2（"timingType"が"interval"かつ"period"が"first","last"の場合）

```
{
  "descriptions": {
    "ja": "エアコンの室内温度",
    "en": "room temperature of the air-conditioner"
  },
  "resourceType": "property",
  "resourceName": "operationStatus",
  "timing": {
    "timingType": "interval",
    "intervalValue": 30,
    "intervalUnit": "min"
  },
  "period": {
    "first": "2019-04-01T08:00:00+09:00",
    "last": "2023-04-24T22:00:00+09:00"
  },
  "maxRequestDuration": {
    "value": 30,
    "unit": "day"
  },
  "maxRequestNumberOfDevices": 100,
  "retrieveMethod": "async",
  "dataStructure": ["device", "time"]
}
```

■ レスポンス

Property	Type	Required	Description
descriptions	object	Yes	登録済みの groupHistory に関する記述
descriptions.ja	string	Yes	登録済みの groupHistory の内容説明（日本語）

Property	Type	Required	Description
descriptions.en	string	Yes	登録済みの groupHistory の内容説明（英語）
resourceType	string	Yes	複数機器履歴対象リソースの種類。現時点でのサポートはプロパティリソース "property"のみ
resourceName	string	Yes	複数機器履歴対象リソース名。履歴対象リソースの種類に対応し、取得対象となる複数機器履歴データのリソース名
timing	object	Yes	複数機器履歴データの取得タイミング
timing.timingType	string	Yes	複数機器履歴データの取得タイミングのタイプ。"onChange", "interval"のいずれか
timing.intervalValue	Number	No	時間間隔値。上記 timingType が "interval"の場合、必須
timing.intervalUnit	string	No	時間間隔単位。"sec", "min", "hour", "day", "month", "year"のいずれか。上記 timingType が "interval"の場合、必須
period	object	Yes	複数機器履歴データの取り扱い期間
period.duration	object	No*1	現時点を起点とした期間情報
period.duration.value	number	No*1	duration の値
period.duration.unit	string	No*1	duration の単位。"day", "month", "year"のいずれか
period.first	string	No*1	複数機器履歴データの取得開始時刻。RFC3339（ISO 8601）準拠
period.last	string	No*1	複数機器履歴データの取得最新時刻。RFC3339（ISO 8601）準拠。時間経過に伴い更新されても良い
maxRequestDuration	object	Yes	取得可能（from, to で設定可能）な期間情報
maxRequestDuration.value	number	Yes	maxRequestDuration の値

Property	Type	Required	Description
maxRequestDuration.unit	string	Yes	maxRequestDuration の単位。"day", "month", "year"のいずれか
maxRequestNumberOfDevices	number	Yes	複数機器履歴データで取得可能な最大要求台数
retrieveMethod	string	Yes	複数機器履歴データ取得方法。"sync", "async", "file"のいずれか
dataStructure	array[string]	Yes	履歴データのデータ構造を列挙する。値は "device", "time" なお、履歴データのデータ構造に関しては、以下の API で記述する POST /groupHistories/<group history id>/actions/retrieveBySync

*1：period の表現方法を以下に示す

Type	period.first	period.last	period.duration	description
1	-	-	✓	時刻（現在時刻 - duration）から現在まで
2	✓	-	-	時刻 "first" から現在まで
3	✓	✓	-	時刻 "first" から 時刻 "last" まで

POST /groupHistories/<group history id>/actions/prepareRetrieveDataBySync

クライアントは、group history ID にて指定される複数機器履歴データセットのうち、所望する複数機器履歴データの機器のリスト（groupId, deviceId）、対象期間（開始時刻 "from"、終了時刻 "to"）、dataStructure を指定し、サーバに対して取得対象となる複数機器履歴データサブ セットの確定を要求する（対象期間はオプション）。サーバは複数機器履歴データサブセットを返却できる準備が整ったら、クライアントへ data ID（"dataId"）と総件数（"count"）、1 応答あたりの件数 （"countPerPage"）、タイムアウト時刻（"expirationTime"）を返却する。

グループを指定する groupId、もしくは機器を指定する deviceId は少なくとも 1 つは指定することとする。なお、groupId、deviceId ともに複数を指定することも可能とする。また、全機器を指定する場合、deviceId の値を"all"と指定する。

クライアントは、groupHistory ID にて指定される履歴データセットのうち、所望する履歴データの対象期間（開始時刻 "from"、終了時刻 "to"）を指定し、サーバに対して取得対象となる履歴データサブセットの確定を要求する（対象期間はオプション）。histories における prepareRetrieveData アクションと同様の処理内容となる。

クライアントが指定する `dataStructure` をサーバ側でサポートしていない場合、HTTP ステータスコード 400 でエラー応答とする。

■ リクエスト定義

```
{
  "groupId": [<group id>],
  "deviceId": [<device id>],
  "from": <time stamp>,
  "to": <time stamp>,
  "dataStructure": "device" | "time"
}
```

■ リクエスト例

```
{
  "groupId": [],
  "deviceId": ["device-001", "device-002", "device-003"],
  "from": "2023-08-21T00:00:00+09:00",
  "to": "2023-08-23T00:00:00+09:00",
  "dataStructure": "device"
}
```

■ レスポンス定義

```
{
  "dataId": <data id>,
  "count": <count number>,
  "countPerPage": <count per page>,
  "expirationTime": <time stamp>
}
```

■ レスポンス例

```
{
  "dataId": "0023",
  "count": 10000,
  "countPerPage": 5000,
  "expirationTime": "2023-12-31T10:00:00+09:00"
}
```

リクエスト：

Property	Type	Required	Description
groupId	array[string]	Yes※	対象機器を指定するための group ID を列挙する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合はエラーとなる
deviceId	array[string]	Yes※	対象機器を指定するための device ID を列挙する。全ての機器を指定する場合は予約語 ″all″ を用いて、"deviceId" : ["all"] と記述する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合は、エラーとなる
from	string	No	検索開始時刻。RFC3339（ISO 8601）準拠
to	string	No	検索終了時刻。RFC3339（ISO 8601）準拠
dataStructure	string	Yes	複数機器履歴データのデータ構造を ″device″ または ″time″ を設定して指定する

※groupId、deviceId の一方の指定を必須とし、指定しなかったプロパティについては、リクエストボディにプロパティを設定しなくてよい。(設定する場合の値は、空配列（empty array）とする)

レスポンス：

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID。実行時に割り当てられる。一定時間（=expirationTime、サーバ規定）経過後に自動で削除される
count	number	Yes	取得対象の複数機器履歴データサブセット内の履歴データ総件数
countPerPage	number	Yes	1 応答（page）あたりに返却可能な履歴データの件数。サーバが一度に応答できる履歴データの最大数（単位）となる
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed）となる

POST /groupHistories/<group history id>/actions/retrieveBySync

クライアントは、data ID で指定される複数機器履歴データサブセットの取得を実行する。

複数機器履歴データサブセット内の履歴データの総件数が、1 応答あたりに返却可能な履歴データの件数を上回る場合は、複数機器履歴データサブセットは複数の応答（ページ）に分割される。クライアントから複数機器履歴データサブセットを取得するには、data ID に加え、ページ番号を指定する必要がある。ただし、1 ページ目についてはページ番号の指定を省略することができる。

レスポンスでは、サーバでの処理状態（"processStatus"）が返却され、成功終了時（"succeeded"）には、これに加え、複数機器履歴データ（"data"）がページ単位にて返却される。複数機器履歴データは deviceId、取得時刻（"time"）および取得値（"value"）の組の配列から構成される。取得値はリソース種（"resourceType"）・リソース名"resourceName"）に応じて、様々なデータ型をとりうる。dataStructure が "device" の場合と、"time" の場合で配列が異なる。

最終ページを超えるページ番号を指定した場合は、HTTP ステータスコード 404(Not Found)を返す。 "processStatus"は、本操作で返却される複数機器履歴データのサーバ処理状態を示す。複数機器履歴データを含む応答が返却される場合は"succeeded"、サーバ処理が失敗／タイムアウト（値はサーバにより規定）した場合は"failed"を、いずれも HTTP ステータスコード 200 で返す。 data ID（および対応する複数機器履歴データサブセット）は、サーバ規定による一定期間経過後に解放される。

■ リクエスト定義

```
{
  "dataId": <data id>,
  "page": <page number>
}
```

■ リクエスト例

```
{
  "dataId": "0023",
  "page": 10
}
```

■ レスポンス定義 1（並び順がデバイス(dataStructure = device)の場合）

```
{
  "porocessStatus": "succeeded"|"failed"|"aborted",
  "resourceType": "property",
  "resourceName": <resource name>,
  "expirationTime": <time stamp>,
  "data": <array data>,
```

```
  "data[].deviceId": <device ID>,
  "data[].deviceData": <array data>,
  "data[].deviceData[].time": <time stamp>,
  "data[].deviceData[].value": <recorded resource value>
}
```

■ レスポンス定義 2（並び順が日時(dataStructure = time)の場合）

```
{
  "porocessStatus": "succeeded"|"failed"|"aborted",
  "resourceType": "property",
  "resourceName": <resource name>,
  "expirationTime": <time stamp>,
  "data": <array data>,
  "data[].time": <time stamp>,
  "data[].deviceId": <device ID>,
  "data[].value": <recorded resource value>
}
```

■ レスポンス例 1（成功時、並び順がデバイス(dataStructure = device)の場合）

```
{
  "porocessStatus": "succeeded",
  "resourceType": "property",
  "resourceName": "normalDirectionCumulativeElectricEnergy",
  "expirationTime": "2023-09-01T10:00:00+09:00",
  "data": [
    {
      "deviceId": "device-001",
      "deviceData": [
        { "time": "2023-08-21T10:00:00+09:00", "value": 100 },
        { "time": " 2023-08-21T10:30:00+09:00", "value": 105 },
        ...
      ]
    },
    {
      "deviceId": "device-002",
      "deviceData": [
        { "time": " 2023-08-21T10:00:00+09:00", "value": 150 },
        { "time": " 2023-08-21T10:30:00+09:00", "value": 155 },
        ...
      ]
    },
    ...
```

```
    ]
  }
```

■ レスポンス例 2（成功時、並び順が日時(dataStructure = time)の場合）

```
{
  "porocessStatus": "succeeded",
  "resourceType": "property",
  "resourceName": "normalDirectionCumulativeElectricEnergy",
  "expirationTime": "2023-09-01T10:00:00+09:00",
  "data": [
    {
      "time": "2023-08-21T10:00:00+09:00 ",
      "deviceId": "device-001",
      "value": 95
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-002",
      "value": 200
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-003",
      "value": 852
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-004",
      "value": 1005
    },
    ...
  ]
}
```

■ レスポンス例 3（サーバ処理失敗またはタイムアウト）

```
{
  "porocessStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID。実行時に割り当てられる。一定時間（=expirationTime、サーバ規定）経過後に自動で削除される
page	string	No	複数機器履歴データサブセットの分割されたページ番号。1 ページ目については省略可能

レスポンス 1（並び順がデバイス(dataStructure = device)の場合）:

Property	Type	Required	Description	*1	*2
processStatus(*)	string	Yes	・サーバ側の複数機器履歴データサブセット準備状況を表す。 ・状況は 2 つに分類される（"succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了)）	○	○
resourceType	string	Yes	複数機器履歴データのリソース種。プロパティリソース “property"で固定（将来的に は"action"、"event"についても検討 予定）	○	-
resourceName	string	Yes	複数機器履歴データのリソース名	○	-
data	array	Yes	・取得対象の複数機器履歴データサブセット内の履歴データは通常 array 型で返却する。総件数が 0 の場合は空の[]となる ・dataStructure に応じて複数機器履歴データのフォーマットが異なる	○	-
data[].deviceId	string	No	履歴対象リソースの device ID	○	-
data[].deviceData	array	No	device ID に対応する履歴データ	○	-
data[].deviceData[].time	string	No	取得時刻。RFC3339（ISO 8601）準拠	○	-

Property	Type	Required	Description	*1	*2
data[].deviceData[].value	*3	No	取得値。データ欠損している場合は省略される（"time"は省略されない）	○	-
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。 RFC3339（ISO 8601）準拠・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed)となる	○	-

※：*1, *2は、processStatusがそれぞれ"succeeded", "failed"の場合。

※：*3は、"string","number","boolean","object","array"のいずれかの型。

レスポンス 2（並び順がデバイス(dataStructure = time)の場合）：

Property	Type	Required	Description	*1	*2
processStatus	string	Yes	・サーバ側の複数機器履歴データサブセット準備状況を表す。 ・状況は 2 つに分類される("succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了))	○	○
resourceType	string	Yes	複数機器履歴データのリソース種。プロパティリソース “property"で固定（将来的には"action"、"event"についても検討 予定）	○	-
resourceName	string	Yes	複数機器履歴データのリソース名	○	-
data	array	Yes	・取得対象の複数機器履歴データサブセット内の履歴データは通常 array 型で返却する。総件数が 0 の場合は空の[]となる ・dataStructure に応じて複数機器履歴データのフォーマットが異なる	○	-
data[].time	string	No	取得時刻。RFC3339（ISO 8601）準拠	○	-
data[].deviceId	string	No	履歴対象リソースの device ID	○	-

Property	Type	Required	Description	*1	*2
data[].value	*3	No	取得値。データ欠損している場合は省略される（"time"は省略されない）	○	-
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。 RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed)となる	○	-

※：*1, *2は、processStatusがそれぞれ"succeeded", "failed"の場合。

※：*3は、"string","number","boolean","object","array"のいずれかの型。

POST /groupHistories/<group history id>/actions/prepareRetrieveDataByAsync

クライアントは、group history ID にて指定される複数機器履歴データセットのうち、所望する複数機器履歴データの機器のリスト（groupId, deviceId）、対象期間（開始時刻 "from"、終了時刻 "to"）、dataStructure を指定し、サーバに対して取得対象となる複数機器履歴データサブセットの確定を要求する（対象期間はオプション）。サーバはクライアントへ data ID（"dataId"）、データサブセット作成の完了推定時刻（"estimatedTimeOfCompletion"）、タイムアウト時刻（"expirationTime"）を返却する。

グループを指定する groupId、もしくは機器を指定する deviceId は少なくとも 1 つは指定することとする。なお、groupId、deviceId ともに複数を指定することも可能とする。また、全機器を指定する場合、deviceId の値を"all"と指定する。

クライアントは、groupHistory ID にて指定される履歴データセットのうち、所望する履歴データの対象期間（開始時刻 "from"、終了時刻 "to"）を指定し、サーバに対して取得対象となる履歴データサブセットの確定を要求する（対象期間はオプション）。histories における prepareRetrieveData アクションと同様の処理内容となる。

クライアントが指定する dataStructure をサーバ側でサポートしていない場合、HTTP ステータスコード 400 でエラー応答とする。

■ リクエスト定義

```
{
  "groupId": [<group id>],
  "deviceId": [<device id>],
  "from": <time stamp>,
  "to": <time stamp>,
  "dataStructure": "device" | "time"
}
```

■ リクエスト例

```
{
  "groupId": [],
  "deviceId": ["all"],
  "from": "2023-08-21T00:00:00+09:00",
  "to": "2023-09-21T00:00:00+09:00",
  "dataStructure": "device"
}
```

■ レスポンス定義

```
{
  "dataId": <data id>,
  "estimatedTimeOfCompletion": <time stamp>,
  "expirationTime": <time stamp>
}
```

■ レスポンス例

```
{
  "dataId": "0023",
  "estimatedTimeOfCompletion": "2023-12-31T12:00:00+09:00",
  "expirationTime": "2023-12-31T18:00:00+09:00"
}
```

リクエスト：

Property	Type	Required	Description
groupId	array[string]	Yes※	対象機器を指定するための group ID を列挙する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合はエラーとなる

Property	Type	Required	Description
deviceId	array[string]	Yes※	対象機器を指定するための device ID を列挙する。全ての機器を指定する場合は予約語 "all" を用いて、"deviceId" : ["all"] と記述する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合は、エラーとなる
from	string	No	検索開始時刻。RFC3339（ISO 8601）準拠
to	string	No	検索終了時刻。RFC3339（ISO 8601）準拠
dataStructure	string	Yes	複数機器履歴データのデータ構造を "device" または "time" を設定して指定する

※groupId、deviceId の一方の指摘を必須とし、指定しなかったプロパティについては、リクエストボディにプロパティを設定しなくてよい。(設定する場合の値は、空配列（empty array）とする)

レスポンス：

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID。実行時に割り当てられる。一定時間（=expirationTime、サーバ規定）経過後に自動で削除される
estimatedTimeOfCompletion	string	No	複数機器履歴データサブセット作成が完了するであろう、完了推定時刻。RFC3339（ISO 8601）準拠
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。RFC3339（ISO 8601）準拠 ・履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、履歴データサブセット取得不可状態(processStatus = failed) となる

POST /groupHistories/<group history id>/actions/inquireByAsync

クライアントは、data ID で指定される複数機器履歴データサブセットのサーバにおける取得準備の状況を確認する。サーバはクライアントへ data ID（"dataId"）、本操作で返却される複数機器履歴データのサーバ処理状態（"processStatus"）、総件数（"count"）、1 応答あたりの件数（"countPerPage"）、データサブセット作成の完了推定時刻（"estimatedTimeOfCompletion"）、タイムアウト時刻（"expirationTime"）を返却する。 総件数（"count"）および 1 応答あたりの件数 （"countPerPage"）を返却可能な場合

は"succeeded"、サーバ都合で準備中の場合は"inProgress"、サーバ処理が失敗／タイムアウト（値はサーバにより規定）した場合は"failed"を、いずれも HTTP ステータスコード 200 で返す。

■ リクエスト定義

```
{
  "dataId": <data id>
}
```

■ リクエスト例

```
{
  "dataId": "0023"
}
```

■ レスポンス定義

```
{
  "porocessStatus": "inProgress"|"succeeded"|"failed"|"aborted",
  "count": <count number>,
  "countPerPage": <count per page>,
  "estimatedTimeOfCompletion": <time stamp>,
  "expirationTime": <time stamp>
}
```

■ レスポンス例（成功時）

```
{
  "porocessStatus": "succeeded",
  "count": 10000,
  "countPerPage": 5000,
  "expirationTime": "2023-12-31T18:00:00+09:00"
}
```

■ レスポンス例（サーバ処理中にて待ち状態）

```
{
  "porocessStatus": "inProgress",
  "estimatedTimeOfCompletion": "2023-12-31T12:00:00+09:00",
```

```
    "expirationTime": "2023-12-31T18:00:00+09:00"
  }
```

■ レスポンス例（サーバ処理失敗またはタイムアウト）

```
{
  "porocessStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID

レスポンス：

Property	Type	Required	Description	*1	*2	*3
processStatus(*)	string	Yes	・サーバ側の複数機器履歴データサブセット準備状況を表す。 ・状況は 3 つに分類される("inProgress"(サーバ処理中), "succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了))	○	○	○
count	number	Yes	取得対象の複数機器履歴データサブセット内の履歴データ総件数	-	○	-
countPerPage	number	Yes	1 応答（page）あたりに返却可能な履歴データの件数。サーバが一度に応答できる履歴データの最大数（単位）となる	-	○	-

Property	Type	Required	Description	*1	*2	*3
estimatedTimeOfCompletion	string	No	複数機器履歴データサブセット作成が完了するであろう、完了推定時刻。 RFC3339（ISO 8601）準拠	○	-	-
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。 RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed) となる	○	○	-

※：*1, *2, *3は、processStatusがそれぞれ"inProgress", "succeeded", "failed"の場合。

POST /groupHistories/<group history id>/actions/retrieveByAsync

クライアントは、「inquireByAsync」にて総件数（"count"）および 1 応答あたりの件数（"countPerPage"）を取得した後に、data ID で指定される複数機器履歴データサブセットの取得を実行する。 複数機器履歴データサブセット内の履歴データの総件数が、1 応答あたりに返却可能な履歴データの件数を上回る場合は、複数機器履歴データサブセットは複数の応答（ページ）に分割される。クライアントから複数機器履歴データサブセットを取得するには、data ID に加え、ページ番号を指定する必要がある。ただし、1 ページ目についてはページ番号の指定を省略することができる。 レスポンスでは、サーバでの処理状態（"processStatus"）が返却され、成功終了時（"succeeded"）には、これに加え、複数機器履歴データ（"data"）がページ単位にて返却される。複数機器履歴データは deviceId、取得時刻（"time"）および取得値（"value"）の組の配列から構成される。取得値はリソース種（"resourceType"）・リソース名（"resourceName"）に応じて、様々なデータ型をとりうる。dataStructure が "device" の場合と、"time" の場合で配列が異なる。dataStructure の具体例は、retrieveBySync を参照のこと。

■ リクエスト定義

```
{
  "dataId": <data id>,
  "page": <page number>
}
```

■ リクエスト例

```
{
  "dataId": "0023",
  "page": 10
}
```

■ レスポンス定義 1（並び順がデバイス（dataStructure=device）の場合）

```
{
  "porocessStatus": "inProgress"|"succeeded"|"failed"|"aborted",
  "resourceType": "property",
  "resourceName": <resource name>,
  "expirationTime": <time stamp>,
  "data": <array data>,
  "data[].deviceId": <device ID>,
  "data[].deviceData": <array data>,
  "data[].deviceData[].time": <time stamp>,
  "data[].deviceData[].value": <recorded resource value>
}
```

■ レスポンス定義 2（並び順が日時（dataStructure=time）の場合）

```
{
  "porocessStatus": "inProgress"|"succeeded"|"failed"|"aborted",
  "resourceType": "property",
  "resourceName": <resource name>,
  "expirationTime": <time stamp>,
  "data": <array data>,
  "data[].time": <time stamp>,
  "data[].deviceId": <device ID>,
  "data[].value": <recorded resource value>
}
```

■ レスポンス例 1（成功時、並び順がデバイス（dataStructure=device）の場合）

```
{
  "porocessStatus": "succeeded",
  "resourceType": "property",
  "resourceName": "normalDirectionCumulativeElectricEnergy",
  "expirationTime": "2023-09-01T10:00:00+09:00",
```

```
  "data": [
    {
      "deviceId": "device-001",
      "deviceData": [
        { "time": "2023-08-21T10:00:00+09:00", "value": 100 },
        { "time": "2023-08-21T10:30:00+09:00", "value": 105 },
        ...
      ]
    },
    {
      "deviceId": "device-002",
      "deviceData": [
        { "time": "2023-08-21T10:00:00+09:00", "value": 150 },
        { "time": "2023-08-21T10:30:00+09:00", "value": 155 },
        ...
      ]
    },
    ...
  ]
}
```

■ レスポンス例 2（成功時、並び順が日時(dataStructure=time)の場合）

```
{
  "porocessStatus": "succeeded",
  "resourceType": "property",
  "resourceName": "normalDirectionCumulativeElectricEnergy",
  "expirationTime": "2023-09-01T10:00:00+09:00",
  "data": [
    {
      "time": "2023-08-21T10:00:00+09:00 ",
      "deviceId": "device-001",
      "value": 95
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-002",
      "value": 200
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-003",
      "value": 852
    },
    {
      "time": "2023-08-21T10:00:00+09:00",
      "deviceId": "device-004",
```

```
        "value": 1005
      },
      ...
    ]
  }
```

■ レスポンス例 3（サーバ処理中にて待ち状態）

```
{
  "porocessStatus": "inProgress",
  "estimatedTimeOfCompletion": "2023-08-31T10:00:00+09:00",
  "expirationTime": "2023-09-01T10:00:00+09:00"
}
```

■ レスポンス例 4（サーバ処理失敗またはタイムアウト）

```
{
  "porocessStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID
page	number	No	複数機器履歴データサブセットの分割されたページ番号。1 ページ目については省略可能

レスポンス：

Property	Type	Required	Description	*1	*2	*3
processStatus(*)	string	Yes	・サーバ側の複数機器履歴データサブセット準備状況を表す。 ・状況は 3 つに分類される("inProgress"(サーバ処理中), "succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了))	○	○	○

Property	Type	Required	Description	*1	*2	*3
resourceType	string	Yes	複数機器履歴データのリソース種。プロパティリソース“property”で固定（将来的には"action"、"event"についても検討 予定）	-	○	-
resourceName	string	Yes	複数機器履歴データのリソース名	-	○	-
estimatedTimeOfCompletion	string	No	複数機器履歴データサブセット作成が完了するであろう、完了推定時刻。RFC3339（ISO 8601）準拠	○	-	-
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed) となる	○	○	-
data	array	Yes	・取得対象の複数機器履歴データサブセット内の履歴データは通常 array 型で返却する。総件数が 0 の場合は空の[]となる ・retrieveBySync レスポンス定義における data プロパティと同じデータ構造をとる	-	○	-

※：*1, *2, *3は、processStatusがそれぞれ"inProgress", "succeeded", "failed"の場合。

POST /groupHistories/<group history id>/actions/prepareRetrieveDataByFile

クライアントは、group history ID にて指定される複数機器履歴データセットのうち、所望する複数機器履歴データの機器のリスト（groupId, deviceId）、対象期間（開始時刻 "from"、終了時刻 "to"）、dataStructure を指定し、サーバに対して取得対象となる複数機器履歴データサブセットの確定を要求する（対象期間はオプション）。サーバはクライアントへ data ID（"dataId"）、データサブセット作成の完了推定時刻（"estimatedTimeOfCompletion"）、タイムアウト時刻（"expirationTime"）を返却する。

グループを指定する groupId、もしくは機器を指定する deviceId は少なくとも 1 つは指定することとする。なお、groupId、deviceId ともに複数を指定することも可能とする。また、全機器を指定する場合、deviceId の値を"all"と指定する。

クライアントは、groupHistory ID にて指定される履歴データセットのうち、所望する履歴データの対象期間（開始時刻 "from"、終了時刻 "to"）を指定し、サーバに対して取得対象となる履歴データサブセットの確定を要求する（対象期間はオプション）。histories における prepareRetrieveData アクションと同様の処理内容となる。

クライアントが指定する dataStructure をサーバ側でサポートしていない場合、HTTP ステータスコード 400 でエラー応答とする。

■ リクエスト定義

```
{
  "groupId": [<group id>],
  "deviceId": [<device id>],
  "from": <time stamp>,
  "to": <time stamp>,
  "dataStructure": "device" | "time"
}
```

■ リクエスト例

```
{
  "groupId": [],
  "deviceId": ["device-001", "device-002", "device-003"],
  "from": "2023-08-21T00:00:00+09:00",
  "to": "2023-09-21T00:00:00+09:00",
  "dataStructure": "device"
}
```

■ レスポンス定義

```
{
  "dataId": <data id>,
  "estimatedTimeOfCompletion": <time stamp>,
```

```
  "expirationTime": <time stamp>
}
```

■ レスポンス例

```
{
  "dataId": "0023",
  "estimatedTimeOfCompletion": "2023-12-31T12:00:00+09:00",
  "expirationTime": "2023-12-31T18:00:00+09:00"
}
```

リクエスト：

Property	Type	Required	Description
groupId	array[string]	Yes※	対象機器を指定するための group ID を列挙する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合はエラーとなる
deviceId	array[string]	Yes※	対象機器を指定するための device ID を列挙する。全ての機器を指定する場合は予約語 "all" を用いて、"deviceId" : ["all"] と記述する。group ID および device ID で指定した機器数の合計が maxRequestNumberOfDevices を超えた場合は、エラーとなる
from	string	No	検索開始時刻。RFC3339（ISO 8601）準拠
to	string	No	検索終了時刻。RFC3339（ISO 8601）準拠
dataStructure	string	Yes	複数機器履歴データのデータ構造を"device"または"time"を設定して指定する

※groupId、deviceId の一方の指摘を必須とし、指定しなかったプロパティについては、リクエストボディにプロパティを設定しなくてよい。（設定する場合の値は、空配列（empty array）とする）

レスポンス：

Property	Type	Required	Description

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID。実行時に割り当てられる。一定時間（=expirationTime、サーバ規定）経過後に自動で削除される
estimatedTimeOfCompletion	string	No	複数機器履歴データサブセット作成が完了するであろう、完了推定時刻。RFC3339（ISO 8601）準拠
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed）となる

POST /groupHistories/<group history id>/actions/retrieveByFile

クライアントは、data ID で指定される複数機器履歴データサブセットの取得を実行する。 サーバからのレスポンスでは、サーバでの処理状態（"processStatus"）が返却され、成功終了時（"succeeded"）には、これに加え、複数機器履歴データを格納したファイルの宛先（"url"）が返却される。

■ リクエスト定義

```
{
  "dataId": <data id>
}
```

■ リクエスト例

```
{
  "dataId": "0023"
}
```

■ レスポンス定義

```
{
  "porocessStatus": "inProgress"|"succeeded"|"failed"|"aborted",
  "resourceType": "property",
```

```
  "resourceName": <resource name>,
  "url": <url>,
  "estimatedTimeOfCompletion": <time stamp>,
  "expirationTime": <time stamp>
}
```

■ レスポンス例（成功時）

```
{
  "porocessStatus": "succeeded",
  "resourceType": "property",
  "resourceName": "normalDirectionCumulativeElectricEnergy",
  "url": "https://temporary-url-xxx.com/grouphistories/xxx.json",
  "expirationTime": "2023-12-31T18:00:00+09:00"
}
```

■ レスポンス例（サーバ処理中にて待ち状態）

```
{
  "porocessStatus": "inProgress",
  "estimatedTimeOfCompletion": "2023-12-31T12:00:00+09:00",
  "expirationTime": "2023-12-31T18:00:00+09:00"
}
```

■ レスポンス例（サーバ処理失敗またはタイムアウト）

```
{
  "porocessStatus": "failed"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	取得対象の複数機器履歴データサブセット取得用 data ID

レスポンス：

Property	Type	Required	Description	*1	*2	*3

Property	Type	Required	Description	*1	*2	*3
processStatus(*)	string	Yes	・サーバ側の複数機器履歴データサブセット準備状況を表す。 ・状況は 3 つに分類される("inProgress"(サーバ処理中), "succeeded"(サーバ処理成功にて応答), "failed"(サーバ処理が失敗またはタイムアウトにて実行終了))	○	○	○
resourceType	string	Yes	複数機器履歴データのリソース種。プロパティリソース “property"で固定（将来的には"action"、"event"についても検討 予定）	-	○	-
resourceName	string	Yes	複数機器履歴データのリソース名	-	○	-
url	string	Yes	クライアントがファイルにアクセスするためのurl。url が有効期限付きにする場合は、expirationTime を併用することで有効期限を設定可能	-	○	-
estimatedTimeOfCompletion	string	No	複数機器履歴データサブセット作成が完了するであろう、完了推定時刻。RFC3339（ISO 8601）準拠	○	-	-

Property	Type	Required	Description	*1	*2	*3
expirationTime	string	No	・複数機器履歴データサブセット作成完了しデータサブセット保持のタイムアウト時刻。 RFC3339（ISO 8601）準拠 ・複数機器履歴データサブセット作成後一定の時間(タイムアウト時刻)を経過すると、複数機器履歴データサブセット取得不可状態(processStatus = failed) となる	○	○	-

※：*1, *2, *3は、processStatusがそれぞれ"inProgress", "succeeded", "failed"の場合。

クライアントは、サーバから取得した"url"にアクセスすることで、複数機器履歴データを取得する。

POST /groupHistories/<group history id>/actions/abortPrepareRetrieveDataByAsync

prepareRetrieveDataByAsync実行の中断。リクエスト時、ボディにて中断対象となるdata IDを指定する。中断に成功した場合は、同data IDを返す。

■ リクエスト定義

```
{
  "dataId": <data id>
}
```

■ リクエスト例

```
{
  "dataId": "0023"
}
```

■ レスポンス定義

```
{
  "dataId": <data id>
```

```
}
```

■ レスポンス例

```
{
  "dataId": "0023"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	実行時に割り当てられるdata ID

レスポンス：

Property	Type	Required	Description
dataId	string	Yes	実行時に割り当てられるdata ID

POST /groupHistories/<group history id>/actions/abortPrepareRetrieveDataByFile

prepareRetrieveDataByFile実行の中断。リクエスト時、ボディにて中断対象となるdata IDを指定する。中断に成功した場合は、同data IDを返す。

■ リクエスト定義

```
{
  "dataId": <data id>
}
```

■ リクエスト例

```
{
  "dataId": "0023"
}
```

■ レスポンス定義

```
{
  "dataId": <data id>
}
```

■ レスポンス例

```
{
  "dataId": "0023"
}
```

リクエスト：

Property	Type	Required	Description
dataId	string	Yes	実行時に割り当てられるdata ID

レスポンス：

Property	Type	Required	Description
dataId	string	Yes	実行時に割り当てられるdata ID

POST /groupHistories/<group history id>/actions/getDeviceType

groupHistory IDに対応する deviceTypeの一覧を取得する。groupHistory IDは複数の機器の特定のリソースの履歴データに対応する。リソースのproperty名が同一であれば異なる機器の種類(deviceType)の履歴データも扱うことができる（例１：共通項目の faultStatus, 例２：homeAirConditionerや storageBatteryの operationMode）。従って deviceTypeも複数の場合がある。サーバーがあらかじめ機器の種類を特定できない場合は、予約語 "any" をレスポンスする。

■ レスポンス定義

```
{
  "deviceType": [<device type>]
}
```

■ レスポンス例

```
{
  "deviceType": ["homeAirConditioner"]
}
```

```
{
  "deviceType": ["homeAirConditioner", "storageBattery"]
}
```

```
{
  "deviceType": ["any"]
}
```

レスポンス：

Property	Type	Required	Description
deviceType	array[string]	Yes	device typeを列挙する。device typeが特定できない場合は、予約語 "any" をレスポンスする。

POST /groupHistories/<group history id>/actions/getDeviceId

groupHistory IDに対応する機器一覧を取得する。

■ レスポンス定義

```
{
  "devices": [
    {
      "id": <device id>,
      "deviceType": <device  type>,
      "protocol": {
        "type": <ECHONET Lite protocol version>,
        "version": <Appendix release version>
      },
      "manufacturer": {
        "code": <manufacturer code>,
        "descriptions": {
          "ja": <description in Japanese>,
          "en": <description in English>"
        }
      }
    },
    ...
  ]
}
```

■ レスポンス例

```
{
  "devices": [
    {
```

```
        "id": "0xFE000006123456789ABCDEF123456789AB",
        "deviceType": "generalLighting",
        "protocol": {
          "type": "ECHONET_Lite v1.13",
          "version": "Rel.J"
        },
        "manufacturer": {
          "code": "0x000077",
          "descriptions": {
            "ja": "神奈川工科大学",
            "en": "Kanagawa Institute of Technology"
          }
        }
      },
      ...
    ]
  }
```

レスポンス：

Property	Type	Required	Description
devices	array	Yes	機器の基本情報をpropertyとするオブジェクトを列挙する
devices[].id	string	Yes	device ID
devices[].deviceType	string	Yes	device type
devices[].protocol	object	Yes	使用プロトコル
devices[].protocol.type	string	Yes	ECHONET Liteバージョン番号
devices[].protocol.version	string	Yes	Appendixリリース番号
devices[].manufacturer	object	Yes	メーカ情報
devices[].manufacturer.code	string	Yes	メーカコード
devices[].manufacturer.descriptions	object	Yes	メーカ名称
devices[].manufacturer.descriptions.ja	string	Yes	メーカ名称（日本語）
devices[].manufacturer.descriptions.en	string	Yes	メーカ名称（英語）

POST /groupHistories/<group history id>/actions/groupId

groupHistory IDに対応するgroup ID一覧を取得する。

■ レスポンス定義

```
{
  "groups": [
    {
      "id": <group id>,
      "descriptions": {
        "ja": <description in Japanese>,
        "en": <description in English>
      }
    },
    ...
  ]
}
```

■ レスポンス例

```
{
  "groups": [
    {
      "id": "00000011",
      "descriptions": {
        "ja": "EV充放電器で構成されるグループ",
        "en": "A group of evChargerDischargers"
      }
    },
    {
      "id": "00000012",
      "descriptions": {
        "ja": "蓄電池で構成されるグループ",
        "en": "A group of storageBatteries"
      }
    }
  ]
}
```

レスポンス：

Property	Type	Required	Description
groups	array	Yes	group情報をpropertyとするオブジェクトを列挙する
groups[].id	string	Yes	group ID
groups[].descriptions	object	Yes	グループの説明
groups[].descriptions.ja	string	Yes	グループの説明（日本語）

Property	Type	Required	Description
groups[].descriptions.en	string	Yes	グループの説明（英語）

7.6 機器一覧の追加拡張に関する指針

機器一覧取得時の応答内容については、「表 5-2　機器一覧取得時の応答内容」に定義されている内容の変更・削除は行わないことを推奨する。ただし、ベンダ独自に要素を追加することは可能とする。その際の名称は vndを接頭語とする。 「表 5-2　機器一覧取得時の応答内容」に定義されている内容は、以下のように扱う。

deviceType

独自に定義可能とする。ただし、vndを接頭語とする。ECHONET規格のAPPENDIX ECHONET機器オブジェクト詳細規定で定義されている機器でも、「機器仕様部」（別書）で定義されていない機器を独自に定義する場合は vndを接頭語とする。 deviceTypeを定義する場合は、7.6.1も参照のこと。

例： vndLightingWithSensor

protocol

独自に定義可能とする。ただし、type、versionは記述する。

例： "type": "originalProto", "version": "v1.0"

manufacturer

"code": ""（空文字列）とする。descriptionsは独自定義で記述する。

7.7 機器情報（Device Description）の拡張に関する指針

5. 7記載の機器情報（Device Description）に関して、「機器仕様部」（別書）に定義されている機器情報は、名称変更せず利用することを推奨する。各機器が対応外のプロパティ・プロパティ値・アクションを削除するのは構わない。

なお、新たな機器を追加したい、または既存の機器情報に対して新たなプロパティを追加したい場合、機器情報をベンダ独自で拡張しても良い。将来、機器仕様部の定義が拡張された場合に名称が重複しないよう、機器情報をベンダ独自で追加拡張する場合の指針について下記にまとめる。

7.7.1 新たな機器の追加

「機器仕様部」（別書）で定義されていないECHONET Lite機器、4. 4記載の仮想的な機器、ECHONET Lite以外の方式の機器など、「機器仕様部」（別書）で定義されていない機器を新規に定義してもよい。

新たな機器を追加する場合、deviceTypeの名前の先頭に vndを付ける。プロパティやアクションの名前の先頭に vndを付ける必要はない。ECHONET規格のAPPENDIX ECHONET機器オブジェクト詳細規定に対応する定義がない場合は、eojと epcは省略可とする。

7.7.2 既存の機器情報に対する拡張

「機器仕様部」（別書）で既に定義されている機器情報に対して変更を行う場合、以下3つの条件のいずれかに該当する場合は、ベンダ独自の拡張として扱うことが望ましい。

1. 既存のプロパティやアクションと異なる機能のプロパティやアクションを追加する場合。機器仕様部」（別書）で定義済みの機器に対して、新たなプロパティやアクションを追加する場合は、プロパティやアクションの名前の先頭に vndを付けた名前とする。ECHONET規格のAPPENDIX ECHONET機器オブジェクト詳細規定に対応する定義がない場合は、epcは省略可とする。

   例：vndYUV（一般照明クラスならRGB）

2. 既存のプロパティやアクションと同じ名前のプロパティやアクションを定義しているが、共通のプロパティ値が全く存在しないスキーマ定義になっている場合。ただし、1つでもプロパティ値が重複している場合は、本条件に該当しない。

   例：既存定義の typeが numberで、独自定義の typeがstring1. と同様に、プロパティやアクションの名前の先頭に vndを付けた名前とする。

3. 既存の列挙型（enum）のプロパティ値に列挙子を追加する場合。「機器仕様部」（別書）で定義済みの列挙型（enum）のプロパティ値に対して、新たな列挙子を追加する場合は、vndを接頭語とする。

   例： 運転モードの "enum":["auto", "cooling", "heating"]
   に、″vndExtremeCooling″ の列挙子を追加

property objectの内容については、「表 5-4　Property Objectの内容」に定義されている内容の変更は行わないことを推奨する。ただし、ベンダ独自に要素を追加することは可能とする。その際の名称は vndを接頭語とする。

例：vndOriginalProtoCode

「表 5-4　Property Objectの内容」に定義されている内容は、以下のように扱う。

epc

省略可（ECHONET Lite以外の機器においては不要）

epcAtomic

省略可（ECHONET Lite以外の機器においては不要）

descriptions

独自定義で記述する。

urlParameters

独自定義で記述可。ただしJSON Schema形式で記述する。

schema

独自定義で記述可。ただしJSON Schema形式で記述する。

7.8 定義済み機器の組み合わせ使用に関する指針

定義済みの機器を複数組み合わせ、1つの機器として使用する場合、以下に記す指針に基づいて定義することを推奨する。

7.8.1 deviceType名に関する指針

定義済み機器の deviceType名をアルファベット降順に羅列し、lowerCamelにて定義する。なお、deviceType名のセパレータとして"_"を使用することを推奨する。

【定義例】 温度センサ（temperatureSensor）、CO2センサ（co2Sensor）を組み合わせる場合

```
"deviceType" : "co2Sensor_temperatureSensor"
```

7.8.2 プロパティ名に関する指針

組み合わせ定義する機器内でプロパティ名称の重複を避けるため、組み合わせに用いる機器のプロパティの中で、必要なプロパティについては先頭に deviceType名を付したプロパティ名とする。

【定義例】 deviceType ："temperatureSensor"、プロパティ名："value"の場合 プロパティ名 ："temperatureSensorValue"

7.8.3 Device Description定義に関する指針

プロパティの引用元である機器を明確にするため、各プロパティに対して機器の EOJの記載を推奨する。また、EOJを持たない定義済み機器の場合には deviceTypeの記載としてもよい。記載位置は各プロパティの定義の中とし、keyは EOJには"baseEoj “を、deviceTypeには"baseDeviceType"を使用する。（EOJを持つ機器について deviceTypeを併記してもよい。）
なお、プロパティの列挙順は任意とし、定義済み内容をプロパティ名のみ7.6.2に示した指針に従い変更し記載する。

【定義例1】 組み合わせる機器のすべてが EOJを持つ場合

```
{
    "deviceType":"co2Sensor_temperatureSensor",
    "descriptions": {
        "ja": "CO2 温度センサ",
        "en": "CO2 Temperature sensor"
        },
        "properties": {
            "co2SensorValue": {
                "baseEoj": "0x001B",
                "baseDeviceType": "co2Sensor",  ←省略可
                "epc": "0xE0",
```

```
                "descriptions": {
                    "ja": "CO2濃度計測値",
                    "en": "Measured value of CO2 concentration"
                },
                ・・中略・・
                },
                "temperatureSensorValue": {
                    "baseEoj": "0x0011",
                    "baseDeviceType": "temperatureSensor", ←省略可…以下略
```

【定義例2】 組み合わせる機器にEOJを持たないものがある場合

```
{
    "deviceType":"co2Sensor_vndPm25Sensor",
    "descriptions": {
        "ja": "CO2 PM2.5センサ",
        "en": "CO2 PM2.5 sensor"
    },
    "properties": {
        "co2SensorValue": {
            "baseEoj": "0x001B",
            "baseDeviceType": "co2Sensor",←省略可
            "epc": "0xE0","descriptions": {
                "ja": "CO2濃度計測値",
                "en": "Measured value of CO2 concentration"
            },  ・・中略・・
        },
    "vndPm25SensorValue": {
        "baseDeviceType": "vndPm25Sensor", …以下略
```

7.9 echoCommand

WebAPI を利用するサービスの中には、ECHONET Lite で定義された EOJ, ESV, EPC, EDT の値を利用したい場合も考えられる。このような要望に対応するため、下表に示す API を定義する。EOJ, ESV, EPC, EDT の値を利用するコマンドを echoCommand と呼ぶ。

http method	path	Description
GET	/nodes	node profile 情報のリスト取得
POST	/nodes/	echoCommand の送信

GET /nodes

家庭内 LAN に接続されている ECHONET Lite 機器の node profile の情報（識別番号とインスタンスリスト）を "id" と 'instances" として取得する。

■レスポンス定義

```
{
    "nodes": [
       {
           "id":<node profileの識別番号>,
           "instances": [
               {
                   "eoj":<機器のEOJ>
               }
           ]
       },
       ...
   ]
}
```

■レスポンス例

```
{
    "nodes": [
       {
           "id": "FE...0001",
           "instances": [
               {
                   "eoj": "0x013001"
               }
           ]
       },
       {
           "id": "FE...0002",
           "instances": [
               {
                   "eoj": "0x026B01"
               },
               {
                   "eoj": "0x028101"
               },
               {
                   "eoj": "0x028201"
               }
           ]
       }
   ]
}
```

レスポンス：

Property	Type	Required	Description
node	array	Yes	node profile 情報のリスト
nodes[].id	string	Yes	node profile の識別番号（EPC:0x83）
nodes[].instances	array	Yes	インスタンスリスト
nodes[].instances[].eoj	string	Yes	デバイスオブジェクトの EOJ

POST /nodes/<node id>

GET /nodes で取得した"id"の値を <node id>に指定し、JSON データで記述したechoCommand を body dataとして送信する。echoCommandの各要素の値は、16進数表記の文字列（例："0x80", "0xA0"）で記述する。 OPC, PDCはechoCommandの内容から計算で求められるので記述しない。EDTは可変長データなので、"edt" の値はEDTの1バイトごとの値を要素とする配列で記述する（例：0x123456 は ["0x12", "0x34", "0x56"]）。"deoj"はリクエストのbody dataのみ必要である。"seoj"はレスポンスのbody dataのみ必要である。

■リクエスト定義

```
{
    "echoCommand": {
       "deoj":<DEOJ>,
       "esv":<ESV>,
       "operations": [
           {
               "epc":<EPC>,
               "edt": [<EDT DATA>
               ]
           }       },
       ...
    ]
}
```

■レスポンス定義

```
{
   "echoCommand": {
       "seoj":<SEOJ>,
       "esv":<ESV>,
       "operations": [
           {
               "epc":<EPC>,
               "edt": [<EDT DATA>
               ]
```

```
            }
        },
        ...
    ]
}
```

■リクエスト例：エアコンの動作状態取得

```
{
   "echoCommand": {
       "deoj": "0x013001",
       "esv": "0x62",
       "operations": [
           {
               "epc": "0x80"
           }
       ]
   }
}
```

■レスポンス例：エアコンの動作状態取得

```
{
   "echoCommand": {
       "seoj": "0x013001",
       "esv": "0x72",
       "operations": [
           {
               "epc": "0x80",
               "edt": [
                   "0x30"
               ]
           }
       ]
   }
}
```

■リクエスト例：エアコンの動作状態をOFFに設定する

```
{
    "echoCommand": {
       "deoj": "0x013001",
       "esv": "0x61",
```

```
        "operations": [
            {
                "epc": "0x80",
                "edt": [
                    "0x31"
                ]
            }
        ]
    }
}
```

■レスポンス例：エアコンの動作状態をOFFに設定する

```
{
    "echoCommand": {
        "seoj": "0x013001",
        "esv": "0x71",
        "operations": [
            {
                "epc": "0x80"
            }
        ]
    }
}
```

リクエスト：

Property	Type	Required	Description
echoCommand	object	Yes	echoCommand object
echoCommand.deoj	string	Yes	DEOJ
echoCommand.esv	string	Yes	ESV
echoCommand.operations	array	Yes	operationのリスト
echoCommand.operations[].epc	string	Yes	EPC
echoCommand.operations[].edt	array	No	EDT (ESV=0x62 の場合は不要)
echoCommand.operations[].edt[]	string	No	EDT の1バイト毎の値

レスポンス：

Property	Type	Required	Description
echoCommand	object	Yes	echoCommand object
echoCommand.seoj	string	Yes	SEOJ

Property	Type	Required	Description
echoCommand.esv	string	Yes	ESV
echoCommand.operations	array	Yes	operationのリスト
echoCommand.operations[].epc	string	Yes	EPC
echoCommand.operations[].edt	array	Yes	EDT
echoCommand.operations[].edt[]	string	Yes	EDT の1バイト毎の値

第8章 おわりに

本書では、宅内のECHONET Liteモデルをクラウドなどのサーバ上にマッピングし、宅内のECHONET Lite対応機器などをWeb API化したサービスを介して外部クライアントから活用可能とするシステムの構築に際し、Web API設計時に考慮すべき各種指針について示した。 本書は前回ガイドラインで記述した機器操作などの基本ユースケースに加え、特に、応用ユースケース（複数命令の一括指示、機器のグルーピング、履歴データ機器など）にフォーカスをあて、提供可能なサービスの幅を広げ、より高機能なサービスを提供できるよう、ガイドラインの策定をはかった。今後、対応機器の種類を順次増やしつつ、応用ユースケースの拡充やサービス事業者向けに活用しやすい豊富な機能や環境について整備を進めていく。

第9章 謝辞

本書を作成するにあたり、神奈川工科大学の藤田裕之氏には、WGへオブザーバ参加いただき、検討・執筆に際し多大なるご協力を賜りました。深謝いたします。また、議論に参加いただいた関係各位に感謝の気持ちと御礼を申し上げます。

本書を作成するにあたり、神奈川工科大学の藤田裕之氏には、WGへオブザーバ参加いただき、検討・執筆に際し多大なるご協力を賜りました。深謝いたします。また、議論に参加いただいた関係各位に感謝の気持ちと御礼を申し上げます。