第９部　ECHONET ゲートウェイ仕様

改定履歴

・Version1.0　２０００年３月１８日　制定，コンソーシアム会員内公開。
　２０００年７月　一般公開。
・Version1.01　２００１年５月２３日　コンソーシアム会員内公開。
・Version2.00 draft　２００１年８月７日　コンソーシアム会員内公開。
・Version2.01　２００１年 12 月 19 日　コンソーシアム会員内公開。
・Version2.10Preview　２００１年 12 月 28 日　コンソーシアム会員内公開。
・Version2.10Draft　２００2 年 2 月 15 日　コンソーシアム会員内公開。
・Version2.10　２００2 年３月７日　コンソーシアム会員内公開。
・Version2.11　２００２年４月２６日　コンソーシアム会員内公開。
・Version3.00Draft　２００２年６月１２日　コンソーシアム会員内公開。
・Version3.00　２００２年８月２９日　コンソーシアム会員内公開。
・Version3.10Draft　２００２年１１月８日　コンソーシアム会員内公開。
・Version3.10　２００２年１２月１８日　コンソーシアム会員内公開。
・Version3.11　２００３年３月７日　コンソーシアム会員内公開。
・Version3.12　２００３年５月２２日　コンソーシアム会員内公開。
・Version3.20draft　２００３年 10 月 17 日　コンソーシアム会員内公開。

変更のある目次項目は以下の通り

	変更部位（目次項目）	追加・変更概要
1	7.3	7.3 削除

・Version3.20　２００４年　１月　８日　コンソーシアム会員内公開。
・Version3.21　２００４年　５月２６日　コンソーシアム会員内公開。
・Version3.30　２００４年１２月　２日　コンソーシアム会員内公開。
・Version3.40 Draft　２００４年１２月２８日　コンソーシアム会員内公開。
・Version3.40　２００５年　２月　３日　コンソーシアム会員内公開。
・Version3.41　２００５年　５月１１日　コンソーシアム会員内公開。
・Version3.2　２００５年１０月１３日　一般公開。
・Version3.42　２００５年１０月２７日　コンソーシアム会員内公開。
・Version3.50 Draft　２００６年　８月　３日　コンソーシアム会員内公開。

変更のある目次項目は以下の通り

	変更部位（目次項目）	追加・変更概要
1	1.1、1.2	サービスミドルウェアに関する記述を削除
2	旧 1.3、旧 1.5	削除
3	新 1.4	追加
4	第 2 章～第 7 章	削除
5	Part1 全体	追加

・Version3.50　２００６年　９月２０日　コンソーシアム会員内公開。

・Version3.51 Draft	２００７年　２月　２日	コンソーシアム会員内公開。
・Version3.60	２００７年　３月　５日	コンソーシアム会員内公開。
	２００７年１２月１１日	一般公開。
・Version4.00 Draft	２０１１年　４月　７日	コンソーシアム会員内公開。
・Version4.00	２０１１年　６月３０日	コンソーシアム会員内公開。

- エコーネットコンソーシアムが発行している規格類は、工業所有権(特許,実用新案など)に関する抵触の有無に関係なく制定されています。
  エコーネットコンソーシアムは、この規格類の内容に関する工業所有権に対して、一切の責任を負いません。
- 本規格発行者は有償・無償を問わず、いかなる第三者に対しても JAVA、IrDA、Bluetooth®、HBS のライセンスを許諾する権限や免責を与える権限を有していません。JAVA、IrDA、Bluetooth®、HBS を使用する場合、当該使用者は自己の責任と判断に基づき、上記規格について使用許可を得るなどの措置が必要です。
- この書面の使用による、いかなる損害も責任を負うものではありません。

目次

第１章 ECHONET ゲートウェイ仕様概要

１．１ 基本的な考え方

ECHONET は、住宅や中小ビル・店舗などにおいて、設備系システムに接続される機器間の通信プロトコル仕様、および機器内部のインタフェースや処理仕様が主な規格対象範囲としている。しかしながら、住宅や中小ビル・店舗において、設備系システムが孤立して他のシステムとの連携なく動作することは稀であり、宅外の何らかのシステムとの連携や、同一住居内におけるＡＶＣＣ系のシステムとの連携が生じることがほとんどである。ECHONET 規格では、このように外部システムとのインターポイントに位置し、外部システムと ECHONET システムとの連携の中継を司る機器を ECHONET ゲートウェイと呼ぶ。

１．２ ECHONET の外部システムとの接続の考え方

ECHONET における外部システムとの接続の基本的な考え方は以下とする。

（1）宅外からの ECHONET 機器へのアクセスは、ECHONET ゲートウェイにて何らかのセキュリティチェックを行なった後にこれを処理することを基本とする。

（2）ECHONET 通信プロトコルでは ECHONET ドメインを識別するコードは規定しないので、外部システムから複数の ECHONET ドメインを識別したい場合には、アプリケーションソフトウェア独自に各々のドメインを識別する識別子を定義することとする。

１．３ ECHONET ゲートウェイタイプ

ECHONET ゲートウェイは、使用される場面に応じて次の３タイプがあると想定する。このタイプ分けは、確保すべき ECHONET ドメインの情報セキュリティのレベルを想定して行なったものである。

（1）宅外サービスベンダゲートウェイタイプ
（2）外出ユーザゲートウェイタイプ
（3）宅内システムゲートウェイタイプ

（1）宅外サービスベンダゲートウェイタイプ

宅外サービスベンダゲートウェイタイプは、宅外のサービスベンダなどのユーザが用いるシステムと宅内の ECHONET ドメインとが接続される場合を想定したゲートウェイである。すなわち、宅内の ECHONET ドメイン内の情報は、ゲートウェ

イ機器上の情報も含め、その宅内の居住者に管理権があることとし、他人（サービスベンダなどのユーザ）へのこの公開はその居住者が指定する範囲内に限られることを想定したゲートウェイである。

（2）外出ユーザゲートウェイタイプ

外出ユーザゲートウェイタイプは、宅外のシステムと宅内の ECHONET ドメインとが接続される場合を想定したゲートウェイである。ただし、「宅外サービスベンダゲートウェイタイプ」と異なるのは、ユーザが、その宅内の ECHONET ドメイン内の情報の管理権を持つユーザであるという点である。すなわち、本ゲートウェイは、外出しているユーザからアクセスする場合を想定し、所謂テレコントロールを想定したゲートウェイである。

（3）宅内システムゲートウェイタイプ

宅内システムゲートウェイは、ECHONET ドメインが、これと同一の宅内で用いられる他システムと接続される場合を想定したゲートウェイである。すなわち、この場合、他システムのユーザは、ECHONET ドメイン内の情報の管理権を持つユーザと同一であることを想定する。例えば、ＡＶＣＣ系のネットワークとの接続のゲートウェイがある。

１．４ 定義する ECHONET ゲートウェイ

本規格書では、以下に記述する ECHONET ゲートウェイの仕様を規定するものである。

Part 1 : ECHONET-UPnP ゲートウェイ

Part 1.　ECHONET-UPnP ゲートウェイ仕様

本パートでは、ECHONET ゲートウェイのうち、宅内システムゲートウェイタイプで、AV 機器の通信プロトコルである UPnP と接続を行う ECHONET-UPnP ゲートウェイについての詳細を規定する。

第1章 概要

１．１ 基本的な考え方

１．１．１ 開発の背景

近年、情報機器、AV 機器へのネットワーク対応が進んでおり、UPnP™が、その通信プロトコルとして注目され、すでに製品応用が進んでいる。今後、家庭内の ECHONET 機器が情報機器、AV 機器と融合化しシステム化される場合、UPnP 機器とのシステム共存が望まれている。

１．１．２ 規格化の目的

家庭内の ECHONET 機器と AV 機器の相互連携には、ECHONET プロトコルと AV 機器向けの UPnP プロトコルを相互接続するゲートウェイ機能が必要である。そのため、UPnP を具体的なターゲットとしてゲートウェイ機能を規格化することで、ECHONET 機器、AV 機器開発ベンダーの枠を超えて、現実的な相互接続を実現することを目的とする。

１．２ 通信レイヤ上の位置づけ

本書において、UPnP Device Architecture の上位レイヤに位置する ECHONET DCP(Device Control Protocol)を規定する。ECHONET DCP は、ECHONET 用の UPnP コマンドを定義するものである。

ECHONET-UPnP ゲートウェイソフトウェアは、ECHONET 通信処理部及び ECHONET DCP の上位レイヤに位置するものであり、本書（第９部）でその仕様を規定する。本書にて規定する ECHONET DCP、ECHONET-UPnP ゲートウェイソフトウェア部を、図 １．１に網掛けにて示す。

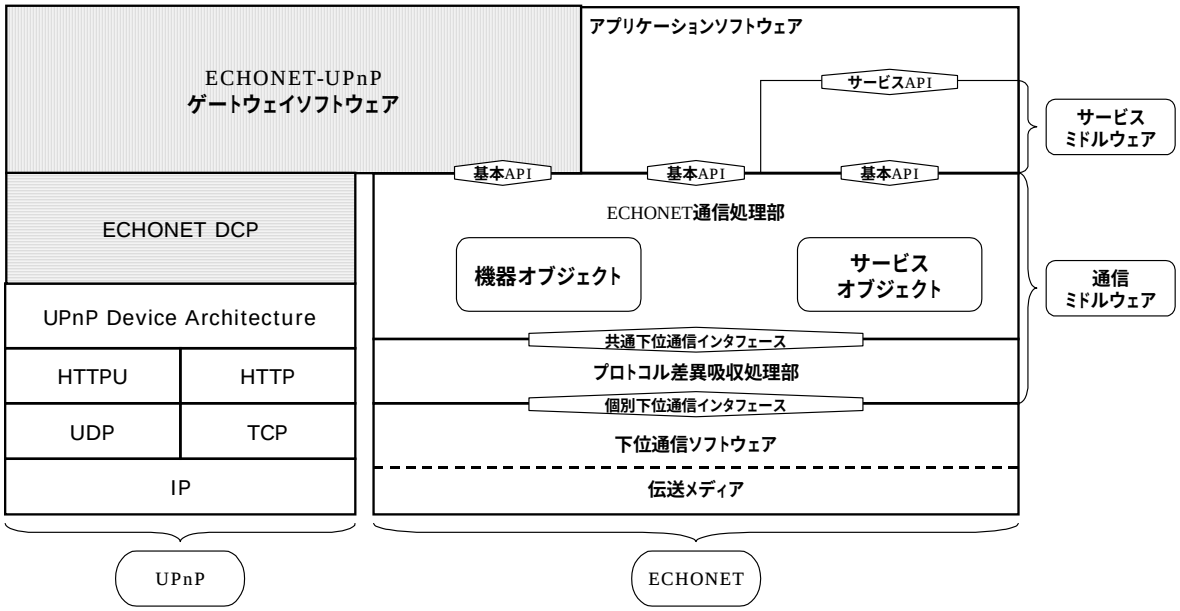

図 １．１　ECHONET-UPnP ゲートウェイソフトウェアの位置づけ

１．３ システム構成と ECHONET-UPnP ゲートウェイの位置づけ

１．３．１ 設計方針

ECHONET-UPnP ゲートウェイソフトウェアの共通化に際しては、以下の設計方針に基づくものとする。

―既存の ECHONET 規格（第９部以外）を変更しない。

―既存の UPnP 仕様（UPnP Device Architecture）を変更しない。

―既存の ECHONET 機器を変更なく収容可能とする。

―新規に ECHONET 機器オブジェクト仕様を追加する際に、対応する UPnP 仕様を簡単に作成できる。

（既存の ECHONET 規格とは、Ver. 3.41 を指す）

また、本章において、UPnP に関する仕様について明示していない部分については、エラーコードを含めて、UPnP Device Architecture の仕様に従うものとする。

１．３．２ システム構成

図 １．２にシステム構成を示す。今回の ECHONET-UPnP ゲートウェイは大きく分類して次に示す２つの方式から構成される。第 1 は、ECHONET 機器の機器オブジェクトを、ECHONET-UPnP ゲートウェイを介して AV 機器上のアプリケーション（UPnP コントロールポイント）から操作する方式である。第 2 は、逆に、AV 機器（UPnP 機器）の機能（UPnP デバイス）を、ECHONET-UPnP ゲートウェイを介して ECHONET 機器から操作する方式である。以下、前者を UPnP デバイス提供方式、後者を ECHONET オブジェクト提供方式と呼ぶ。

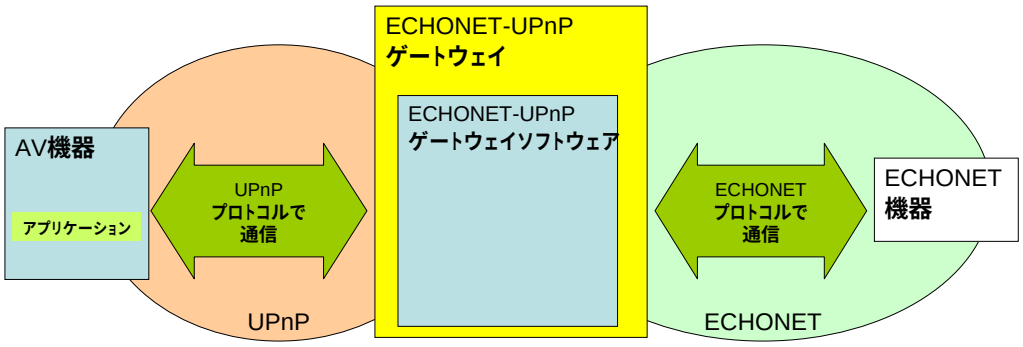

図 １．２　ECHONET-UPnP ゲートウェイシステム構成図

1.3.2.1　UPnP デバイス提供方式

ECHONET 機器の機能を UPnP デバイスとして、UPnP 側へ公開するために、ゲートウェイに ECHONET ノードを制御するための機能、および UPnP デバイス機能を搭載する方式。ゲートウェイは、対象となる ECHONET 機器オブジェクトに対応する UPnP 仮想デバイスを生成し、AV 機器側に搭載された UPnP コントロールポイント（アプリケーション）に機能を提供する方式。また、ECHONET-UPnP ゲートウェイは、ECHONET ノードとして動作するために、第 2 部 9.1 に従い、オブジェクトを搭載することとする。UPnP デバイス提供方式の ECHONET-UPnP ゲートウェイへの搭載は必須とする。

1.3.2.2　ECHONET オブジェクト提供方式

AV 機器の機能を ECHONET オブジェクトとして ECHONET 側へ公開するために、ゲートウェイに他の ECHONET ノードから制御されるための機能、および UPnP コントロールポイント機能を搭載する方式。ゲートウェイは、対象となる UPnP デバイスに対応する ECHONET オブジェクトを生成し、ECHONET システム中の制御を行う ECHONET ノード（アプリケーション）に機能を提供する方式。ECHONET オブジェクト提供方式の ECHONET-UPnP ゲートウェイへの搭載は任意とする。

１．４ 規格化方法

１．４．１ 規格化対象

ECHONET-UPnP ゲートウェイ規格化にあたり、ECHONET プロトコルと UPnP プロトコル間にて、相互に接続された他方の機器を識別・操作可能な技術の仕様を策定する。基本的には、任意のベンダーにより開発された ECHONET-UPnP ゲートウェイが、任意の ECHONET 機器および本規格に基づき開発された UPnP 機器との間の相互接続を実現する。そのための最低限の記述や動作シーケンスを規定する。

(1) プラグアンドプレイ処理の共通化

ECHONET および UPnP における機器のネットワーク接続・脱却時の機器・サービス発見・探索手順は互いに異なる。これを ECHONET-UPnP ゲートウェイにて互いのプロトコル差異を考慮し、プラグアンドプレイ処理可能とする共通仕様を規格化する。

具体的には、あるネットワーク側にて接続を開始する機器を検知すると、他ネットワーク側へその存在を自発的に通知することや、他ネットワーク側の制御端末から送られる機器探索パケットに呼応して応答するといった PnP 処理機能について仕様を規格化する。

(2) ECHONET 機器オブジェクトの UPnP DCP へのマッピングの共通化

UPnP デバイス提供方式に対して策定するものである。UPnP ネットワークから ECHONET 機器を UPnP デバイスとして操作可能にするために、ECHONET 機器オブジェクト（プロパティ）に対応する UPnP の DCP を定義する。ECHONET 規格では現状、すでに数多くの ECHONET 機器オブジェクトが定義されている。また、今後新規の機器オブジェクトも追加される場合も考えられる。それらを考慮し、ECHONET 機器オブジェクト定義を UPnP DCP 定義へ定型的にマッピングするルールを定義する。マッピングルールを定義することで、既存の機器オブジェクトを統一的な記述方法にて表現できるようになる。

具体的には、ECHONET-UPnP ゲートウェイにて ECHONET 機器を仮想的に UPnP 機器として見せることで、UPnP コントロールポイントからの ECHONET 機器への操作を実現する。ECHONET 機器を仮想的な UPnP 機器として見せるためには、ECHONET 機器に対応する device/service description document が必要である。これらの生成に必要な ECHONET 機器情報の選定、ECHONET プロパティコードやサービスコードから、UPnP の device/service description の XML 記述を定型的に生成するためのマッピングルールを策定する。マッピングルール策定には、既存の ECHONET 機器オブジェクトの分類が必要であり、そのための整理を行った上で上記マッピングルールを策定する。また、実際の UPnP コントロールポイントから送信される機器制御コマンドを、ECHONET-UPnP ゲートウェイにて ECHONET コマンドへ変換し、当該機器に送信する処理機能の仕様を規格化する。

(3) UPnP 機器へ情報伝達する ECHONET 機器オブジェクトの共通化

ECHONET オブジェクト提供方式に対して策定するものである。AV 機器の機能を ECHONET 機器オブジェクトとして ECHONET 側へ公開するために必要なオブジェクトの制定を行う。ECHONET 規格として UPnP デバイス側の DCP を策定することはスコープ外とする。

１．４．２ ECHONET 規格化の範囲

図 １．３に、ECHONET-UPnP ゲートウェイとして規格化を行う方法を示す。現在の ECHONET 規格に対し、各 ECHONET 機器オブジェクトを、UPnP のデバイスとして UPnP 側のネットワークに公開するために必要な情報を生成するためのルール（マッピングルール）を策定するのが今回の規格化の中心である。規格化の範囲は以下の通りである。

・ マッピングルールを 3 章以降に定義する。
・ 機器オブジェクトに必要な追加記述、および、現在の ECHONET 機器オブジェクトへの適用例を、Appendix ２に記載する。(今回の作業で新たに追加する部分を、Appendix ２として記述する)
・ 動作シーケンスについては、各方式それぞれについて規定する。

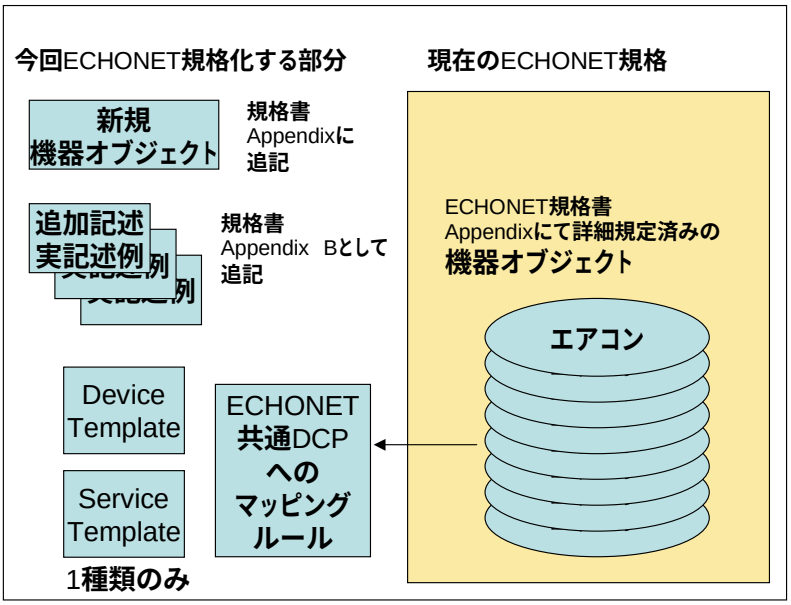

図 １．３ 規格化方法

第２章 用語の定義

- Appendix
  ECHONET 機器オブジェクトのプロパティ構成を詳細に定義したもの
- Appendix ２
  ECHONET 機器オブジェクトのプロパティ構成を UPnP デバイスとして定義したもの
- Appliance 名称
  UPnP ネットワークへ公開する ECHONET 機器オブジェクトの名称
- ECHONET オブジェクト提供方式
  ECHONET-UPnP ゲートウェイが提供する方式の一つ。AV 機器の機能を ECHONET 側へ公開するための方式
- UPnP 仮想デバイス
  ECHONET オブジェクトを UPnP デバイスにマッピングしたもの
- UPnP デバイス提供方式
  ECHONET-UPnP ゲートウェイが提供する方式の一つ。ECHONET 機器の機能を UPnP 側へ公開するための方式
- プロパティタイプ
  ECHONET プロパティをプロパティ値の内容に従って分類したもの
- ECHONET DCP
  ECHONET 機器のための UPnP コマンドを規定した UPnP Device Control Protocol

第３章 UPnP デバイス提供方式

３．１ 基本的な考え方

本章では、ECHONET-UPnP ゲートウェイが UPnP デバイス提供方式にて動作する場合について記述する。本方式のシステム構成例を図 ３．１に示す。

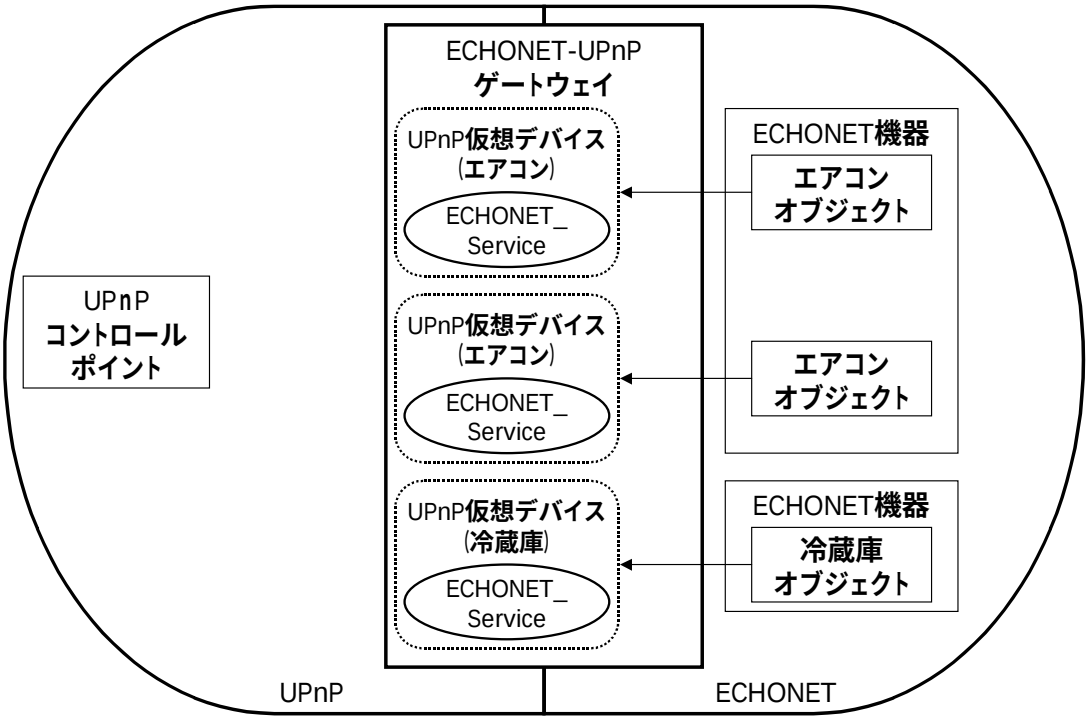

図 ３．１　UPnP デバイス提供方式時のシステム構成例

ECHONET-UPnP ゲートウェイは、UPnP ネットワーク上に ECHONET 機器のサービスを UPnP 仮想デバイスとして提供する。複数の ECHONET オブジェクトを搭載する ECHONET 機器の場合、ECHONET-UPnP ゲートウェイは、各 ECHONET オブジェクトをそれぞれ別の UPnP 仮想デバイスとして提供する。その例を図 ３．１に示す。

ECHONET 機器は、ECHONET アドレス（EA）と ECHONET オブジェクト（EOJ）の組合せによって、特定できる。また、各 UPnP 仮想デバイスに固有なデバイス UUID を ECHONET-UPnP ゲートウェイが設定することによって、UPnP 仮想デバイスは特定できる。従って、ECHONET-UPnP ゲートウェイは、ECHONET ネットワークの ECHONET アドレスと ECHONET オブジェクトの組合せと、UPnP のデバイス UUID をマッピングすることによって、ECHONET 機器と UPnP 仮想デバイスとを１対１に対応させることができる。

UPnP 仮想デバイスのデバイス情報及びサービス情報を公開するための XML Description は、一定の規則に従って作成する。一定の規則は、ECHONET プロパティをタイプ分類したタイプ毎に規定する。新規機器規格時も、規格作成時に ECHONET プロパティをタイプに分類するだけで、ECHONET プロパティから XML Description への定

型的なマッピングが可能である。

UPnP 側へ公開するための機器オブジェクトの名称、ECHONET プロパティの名称、ECHONET データの名称を新規に ECHONET 規格書 Appendix B へ定義する。

３．２ ECHONET プロパティの分類

ECHONET 機器が実行可能なサービスを UPnP ネットワークへ公開する際、ECHONET プロパティを XML Service Description へ定型的にマッピングすることが可能となるための方式を記述する。定型的にマッピングするために、

- XML Service Description の要素構成
- dataType
- VariableName の命名ルール
- Action の命名ルール

を定義する必要がある。「VariableName の命名ルール、Action の命名ルール」は、３．２．１に記述するように ECHONET プロパティタイプに従って分類することによって、マッピングを行うことができる。また、「要素の構成、dataType」は、３．２．２に記述するように ECHONET プロパティのデータ型に従って分類することによって、マッピングを行うことができる。従って、以下の二種類の分類を行う。

- ECHONET プロパティタイプによる分類
- データ型による分類

なお、配列は各要素を独立したプロパティと考えて、分類ルールを適用する。

分類のイメージ図を図 ３．２に示す。

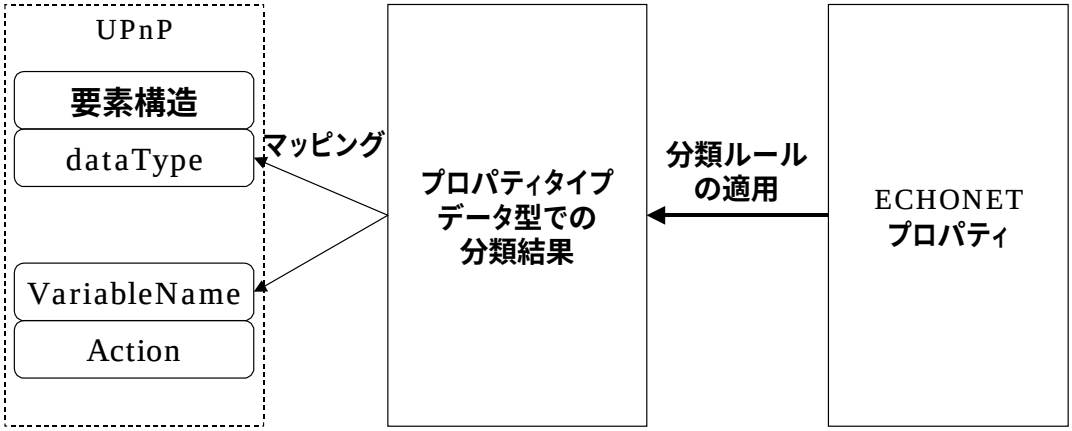

図 ３．２　ECHONET プロパティの分類イメージ図

（1）ECHONET プロパティタイプによる分類

一つ目の分類である「ECHONET プロパティタイプによる分類」は ECHONET プロパティをプロパティ値の内容に従って分類し、ECHONET プロパティごとにプロパティタイプを規定する。プロパティタイプ毎に、VariableName 及び Action の命名ルールを規定する。また、機器を制御するアプリケーションは、規定したプロパティタイプごとにア

プリケーションの雛型を保持することによって、新規機器追加時においても、アプリケーションを自動的に構築することができる。

（2） データ型による分類

二つ目の分類である「データ型による分類」は、ECHONET プロパティのデータ型に基づき ECHONET プロパティを分類する。分類した結果ごとに、XML Service Description へ要素構成、及び dataType を規定する。

新規に ECHONET 機器オブジェクトを追加する場合、従来の ECHONET プロパティの定義とともに、下記に示す分類ルールに従って ECHONET プロパティを分類するものとする。

３．２．１ ECHONET プロパティタイプによる分類

UPnP コントロールポイントでのアプリケーションの構築を考慮して、ECHONET プロパティをプロパティ値の内容に従って分類する。分類した結果をプロパティタイプと定義する。ECHONET プロパティは以下に示す 10 通りのプロパティタイプに分類することができ、その分類は下記に記述する分類ルールに基づくものである。各プロパティタイプの説明を記述する。

（1） 数値型

数値でプロパティ値を示すプロパティである。Data Type は、数値の取り得る範囲によって異なる。具体的には、整数値であるか小数点以下の数値があるか、符号付か符合無しか、データのバイトサイズによって Data Type は決定する。

（2） 日付型

日付を示すデータを保持するプロパティである。Data Type は Date である。XML Description に記述する場合のデータフォーマットは ISO8601 形式に準拠し、yyyy-mm-dd（年－月－日）である。

（3） 時刻型

時刻や時間を保持するプロパティである。Data Type は Time である。XML Description に記述する場合のデータフォーマットは ISO8601 形式に準拠し、hh-mm-ss（時－分－秒）である。

（4） レベル型

数値ではないが、相対的な大小関係や強弱関係を示すデータで ECHONET 機器を制御したり、ECHONET 機器の状態を参照したりするプロパティである。例えば、検知閾値レベルなどがあてはまる。Data Type は String である。

（5） 文字表記型

ECHONET 機器から取得するデータが文字列であったり、ECHONET 機器に制御するデータが文字列であったりするプロパティである。例えば、商品コードや製造番号などが

あげられる。Data Type は String である。

（6） リセット型

定義されているただ一つの値を使用して ECHONET 機器を制御することによって、ある状態をリセットする場合に用いるプロパティである。Data Type は String である。

（7） 切り替え型

二つの値を切り替えることによって ECHONET 機器を制御したり、二つの値のうちいずれかの値を ECHONET 機器から取得したりするプロパティである。例えば、動作状態、異常発生状態などがあげられる。Data Type は String である。

（8） 選択型

三つ以上の値の中から選択することによって ECHONET 機器を制御したり、三つ以上の値の中からいずれかの値を ECHONET 機器から取得したりするプロパティである。例えば、設置場所などがあげられる。Data Type は String である。

（9） 複合型

ECHONET プロパティ内容の値域が、バイトごと、もしくはビットごとに定義しており、複数の項目で構成しているプロパティである。例えば、２バイトずつ定義している定格消費電力や、ビットごとに定義している搭載空気清浄方法などがあげられる。それぞれ（1）数値型～（8）選択型を組み合わせることによって、構成する。

（１０） その他

（1）数値型～（9）複合型のいずれの型にも分類できないプロパティである。ECHONET プロパティ値を文字列変換せず、ECHONET で定義しているバイナリ値を UPnP ネットワークへ公開するものとする。DataType は bin.hex である。

プロパティタイプの一覧を表 ３．１に示す。

表 ３．１ プロパティタイプ一覧

プロパティタイプ	説明
数値型	数値の制御/参照を行う
日付型	日付関連の制御/参照を行う
時刻型	時刻の制御/参照を行う
レベル型	レベルの制御/参照を行う
文字表記型	文字列で表記し、EDT の選択肢がないもの
リセット型	EDT の選択肢１つ、かつアクセスルールは Set のみ
切り替え型	EDT の選択肢２つ
選択型	EDT の選択肢３つ以上
複合型	バイト単位、ビット単位で意味を定義する
その他	上記以外のもの

以下の分類ルールを適用して、ECHONET 規格書 Appendix に記述している ECHONET プロパティの一覧より、ECHONET プロパティを定型的に各プロパティタイプへ分類する。なお分類ルールは、ルール 1 からルール 10 まで規定しているが、ECHONET プロパティの分類は、ルール１からルール 10 まで順番どおりに分類しなければならない。

まず、バイト単位、もしくはビット単位でプロパティデータを定義しており、複数の項目でプロパティデータを構成している物について抽出し、これを「ルール１」とする。ルール１によって抽出した ECHONET プロパティのプロパティタイプを複合型とする。ただし、定義されている項目数が 10 を越えるものに関しては、複合型ではなく、その他とする。なお、複合型の各項目については、以下の数値型～選択型のいずれかの型で分類される。

次に、Appendix の表に単位が記述されているものについて抽出し、これを「ルール２」とする。ルール２によって抽出した ECHONET プロパティのプロパティタイプを数値型とする。

次に、Appendix のプロパティ内容に日付のフォーマットが規定されているものについて抽出し、これを「ルール３」とする。ルール３によって抽出した ECHONET プロパティのプロパティタイプを日付型とする。

次に、Appendix のプロパティ内容に時刻、時間のフォーマットが規定されているものについて抽出し、これを「ルール４」とする。ルール４によって抽出した ECHONET プロパティのプロパティタイプを時刻型とする。

次に、プロパティの内容は数値ではないが、相対的な大小関係や強弱関係を持っているものについて抽出し、これを「ルール５」とする。ルール５によって抽出した ECHONET プロパティのプロパティタイプをレベル型とする。

次に、Appendix の表に単位が記述されていないが、数値として扱っているものについて抽出し、これを「ルール６」とする。ルール６によって抽出した ECHONET プロパテ

ィのプロパティタイプを数値型とする。

次に、設定値の候補の記載がただ一つであるものについて抽出し、これを「ルール 7」とする。ルール 7 によって抽出した ECHONET プロパティのプロパティタイプをリセット型とする。

次に、設定値の候補の記載が二つであるものについて抽出し、これを「ルール 8」とする。ルール 8 によって抽出した ECHONET プロパティのプロパティタイプを切り替え型とする。

次に、設定値の候補の記載が三つ以上であるものについて抽出し、これを「ルール 9」とする。ルール 9 によって抽出した ECHONET プロパティのプロパティタイプを選択型とする。

次に、設定値の候補がなく、プロパティ内容の説明のみ記述されているものについて抽出し、これを「ルール 10」とする。ルール 10 によって抽出した ECHONET プロパティのプロパティタイプを文字表記型とする。

ルール 10 までで分類できないものをその他とする。

上記、記述内容をフローチャートとして、図 ３．３に記述する。

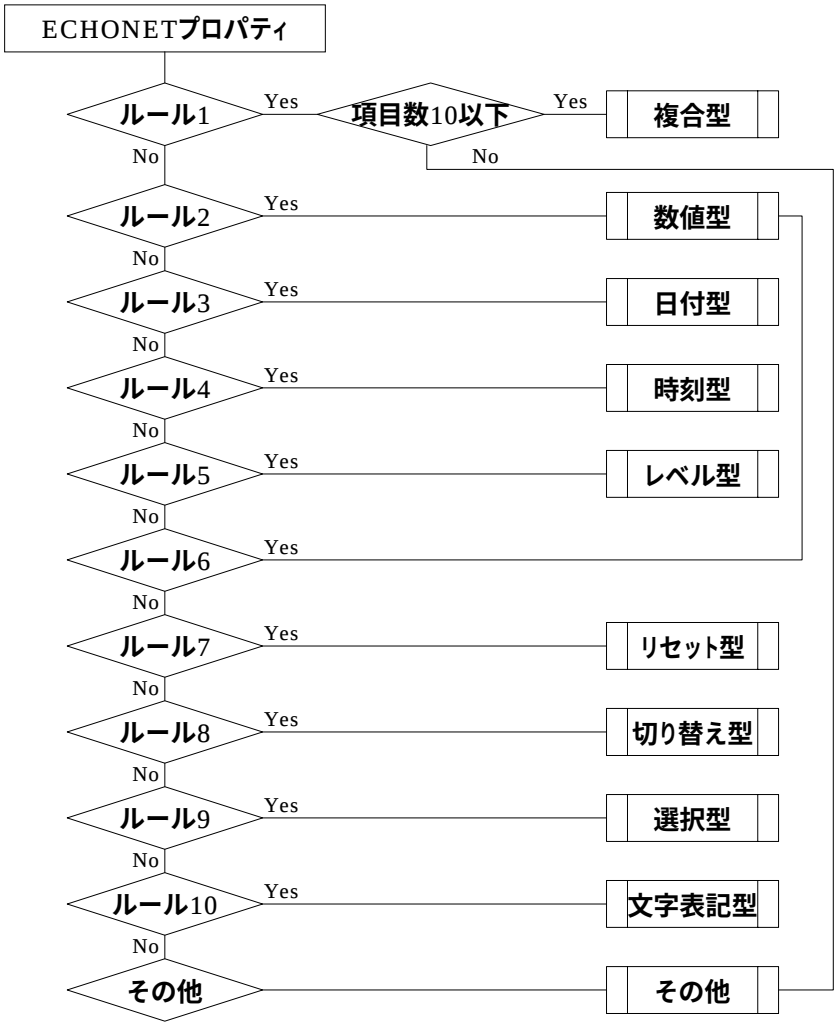

図 ３．３ プロパティタイプによる分類ルール

それぞれの分類ルールについて表 ３．２にまとめる。

表 ３．２ プロパティタイプの分類ルール

ルール	分類する型	内容
ルール１	複合型	・ バイト単位、ビット単位でプロパティデータを定義したもの ・ 定義されている項目数が 10 以下のもの ・ 各項目はルール２以降でプロパティタイプに分類する
ルール２	数値型	・ Appendix の表に単位が記載せれているもの
ルール３	日付型	・ Appendix のプロパティ内容に日付のフォーマットが規定されているもの
ルール４	時刻型	・ Appendix のプロパティ内容に時刻、時間のフォーマットが規定されているもの
ルール５	レベル型	・ プロパティの内容は数値ではないが、相対的な大小関係や強弱関係を持っているもの
ルール６	数値型	・ Appendix の表に単位が記述されていないが、数値として扱っているもの
ルール７	リセット型	・ Appendix のプロパティ内容で設定値の候補の記載がただ一つであるもの
ルール８	切り替え型	・ Appendix のプロパティ内容で設定値の候補の記載が二つであるもの
ルール９	選択型	・ Appendix のプロパティ内容で設定値の候補の記載が三つ以上であるもの
ルール 10	文字表記型	・ Appendix のプロパティ内容で設定値の候補がなく、プロパティ内容の説明のみ記述されているもの

３．２．２ データ型による分類

XML Service Description の記述を考慮して、ECHONET プロパティをデータ型に従って分類する。ECHONET プロパティを以下に示す６通りの型に分類することができ、その分類は下記に記述する分類ルールに基づくものである。分類結果に従って、XML Service Description の要素構成、dataType を決定する。

（１） AVR 型

Appendix に記述している ECHONET プロパティの値域が数値であるもののうち、プロパティ値の範囲を規定しているものを AVR 型とする。

ただし、ECHONET の規格と ECHONET 機器のプロパティ値の範囲は必ずしも一致しているわけではない。また、現バージョンにおいて、プロパティ値の範囲をネットワーク経由で取得することはできない。そのため、本来 AVR 型であるべき ECHONET プロパティに関しても、Value 型で記述してもよいものとする。

XML Service Description に記述する際、serviceStateTable 要素内に allowedValueRange の要素を記述すること。

（2） Value 型

Appendix に記述している ECHONET プロパティの値域が数値であるもののうち、プロパティ値の範囲を規定していないものを Value 型とする。

XML Service Description に記述する際、serviceStateTable 要素内に allowedValueRange、allowedValueList の要素は記述しない。

（3） Date 型

Appendix に記述している ECHONET プロパティの値域が日付、時刻、時間を示す ECHONET プロパティである。

XML Service Description に記述する際、serviceStateTable 要素内に allowedValueRange、allowedValueList の要素は記述しない。

（4） AVL 型

Appendix に記述している ECHONET プロパティの値域が「数値」、「日付、時間」以外であるもののうち、プロパティ値の選択肢を規定しているものを AVL 型とする。

XML Service Description に記述する際、serviceStateTable 要素内に allowedValueList の要素を記述すること。

（5） String 型

Appendix に記述している ECHONET プロパティの値域が「数値」、「日付、時間」以外であるもののうち、プロパティ値の選択肢を規定していないものを String 型とする。

XML Service Description に記述する際、serviceStateTable 要素内に allowedValueRange、allowedValueList の要素は記述しない。

（6） その他型

プロパティタイプがその他型に分類される ECHONET プロパティをその他型とする。

以下の分類ルールを適用して、Appendix に記述している ECHONET プロパティの一覧より、ECHONET プロパティを定型的に分類する。

まず、プロパティタイプがその他型であるものを「その他型」とする。

次に、ECHONET プロパティの内容が「数値」であるものを選択する。さらに数値の範囲の有無で分類する。数値の範囲を規定しているものを「AllowedValueRange 型（以下、AVR 型）」、数値の範囲を規定していないものを「Value 型」とする。ただし、ECHONET の現バージョンでは、ECHONET 機器が動作できる値域をネットワーク経由で取得する

ことはできないので、ECHONET 規格で数値の範囲を規定している ECHONET プロパティについても、Value 型として UPnP へ公開しても良い。

次に、ECHONET プロパティの内容が「日付、時間」関連であるものを選択する。ここで分類した ECHONET プロパティを「Date 型」として記述する。

最後に、残ったプロパティを選択する。さらに残ったプロパティについて、ECHONET プロパティの内容で値域に ECHONET プロパティの選択肢があるものを「AllowedValueList 型（以下、AVL 型)」、選択肢が無いものを「String 型」として規定する。

上記、記述内容をフローチャートとして、図 ３．４に記述する。また、各型の具体的な XML Service Description のフォーマットについては、第６章に記述するものとする。

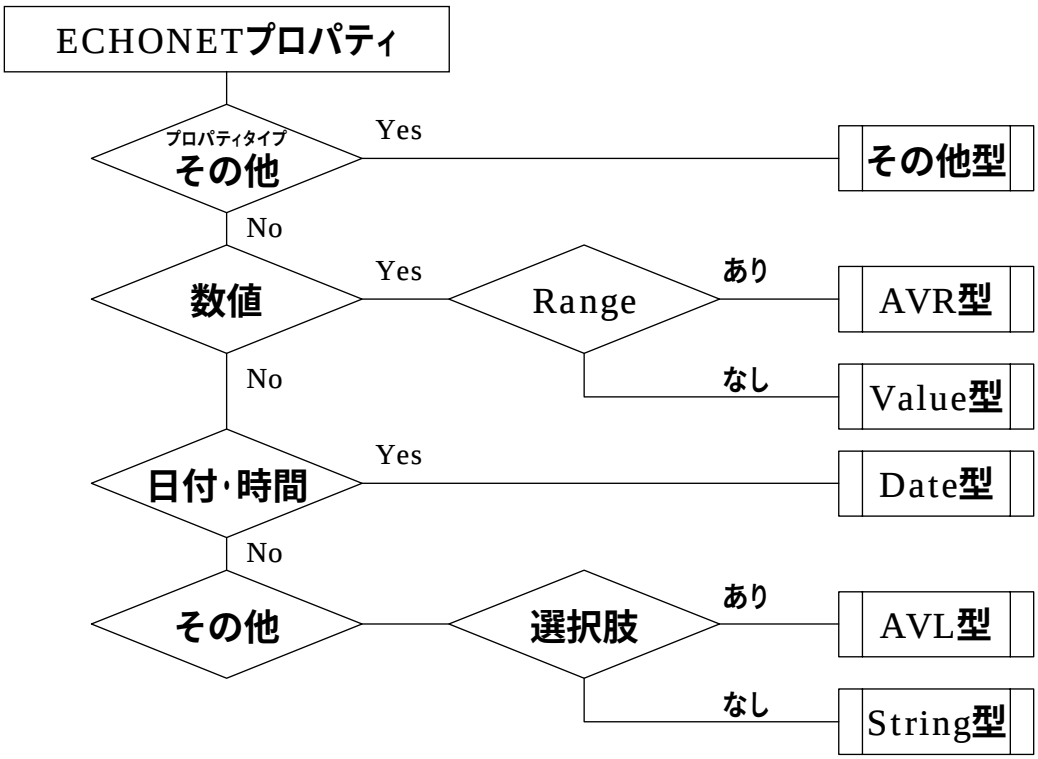

図 ３．４ データ型による分類ルール

３．３ 命名ルール

ECHONET 機器のデバイス情報、及びサービス情報を UPnP ネットワークへ公開するために、ECHONET 機器オブジェクトの名称、ECHONET プロパティの名称、ECHONET データを UPnP と親和性が高い記述方式で記述する必要がある。また、既存及び新規追加の ECHONET プロパティに対する VariableName の名称や Action の名称を簡単に作成するために、現在の ECHONET で規定している名称を利用して定型的に変換する方法が必要である。そのため、以下に示す基本ルールにて記述を生成する。

まず、ECHONET 機器オブジェクトの名称は Device Type の名称に反映させる。また、プロパティタイプによる分類結果と ECHONET プロパティの名称を VariableName の名

称、及びAction の名称へ反映させる。

Action の名称、VariableName の名称、dataType の組合せによって、３．２．１にて分類した ECHONET プロパティタイプを XML Service Description に反映させる。

具体的な命名ルールを以下に記述する。

３．３．１ XML Device Description に必要な命名ルール

XML Device Description を記述する際の ECHONET 機器名称と ECHONET を示すサービスを記述するルールを規定する。

まず、ECHONET 機器を ECHONET_*Appliance* として表すこととし、*Appliance* には ECHONET 機器オブジェクトの名称を記述する。*Appliance* に格納する ECHONET 機器オブジェクトの名称は Appendix の英語版を参照して作成し、Appendix ２に記載することとする。

また、ECHONET_*Appliance* が保持する Service Type を ECHONET_Service とする。

３．３．２ ECHONET プロパティタイプによる分類に基づく命名ルール

VariableName と Action の名称は、ECHONET プロパティによる分類に基づいて規定する。表 ３．３に、ECHONET プロパティタイプ別の命名ルールを示す。

表 ３．３　VariableName と Action の名称ルールリスト

プロパティタイプ	目的語	Action 接頭語
数値型	数値の種別	Write / Read
日付型	Date	Set / Get
時刻型	Time	Set / Get
レベル型	Level	Set / Get
文字表記型	Code	Set / Get
リセット型	–	Reset
切り替え型	Status	Set / Get
選択型	Status	Set / Get
その他	–	Set / Get

VariableName の命名基準としては、

- UPnP の VariableName として違和感が無いこと
- 定型的に命名できること
- 英語として意味が推測できること
- 文字数が Action 接頭語と合わせて 32 文字以内であること
- 非配列型の ECHONET プロパティに関しては、ECHONET プロパティごとに Variable Name を命名すること
- 配列型の ECHONET プロパティに関しては、配列要素ごとに、VariableName を命

名すること

があげられる。よって、以下に示すルールにて作成する。

まず、プロパティタイプ別に、プロパティの内容を示す「目的語」を表 ３．３に記述したように設定する。VariableName は、プロパティ名称を意味のある単語レベルに分割し、前から順番に並べて、表 ３．３に規定した目的語を最後に設定する。これらの命名結果を Appendix ２に記載する。

また、Action の命名ルールは、Action 接頭語に続けて、VariableName を付加することとする。

３．３．３ データ型による分類に基づく dataType の規定

DataType は、データ型による分類に基づいて規定する。

表 3-4 dataType 一覧

型分類	dataType
AVR 型	ui1, ui2, ui4, i1, i2, i4, float
Value 型	ui1, ui2, ui4, i1, i2, i4, float
Date 型	Time / Date
AVL 型	String
String 型	String
その他型	bin.hex

AVR 型、Value 型の dataType については、ECHONET プロパティの値域、データサイズによって判断する。また、Date 型についても同様に ECHONET プロパティの内容から、Data 型もしくは Time 型を判別する。

３．３．４ Argument の命名ルール

XML Service Description にて、アクションに対するパラメータを ArgumentList として定義する。ArgumentList の子要素に Argument が存在するが、その命名ルールを規定する。Argument の命名は 32 文字以内である必要がある。

・ 複合型以外の ECHONET プロパティ

機器を制御する場合、「New*VariableName*」とし、機器の状態を参照する場合、「Current*VariableName*」とする。"*VariableName*"は、該当する VariableName が入る。例えばVarialbeName が OperationStatus である ECHONET プロパティを制御する場合、Argument の命名は、NewOperationStatus となる。

・ 複合型の ECHONET プロパティ

機器を制御する場合、「New *形容詞 VariableName*」とし、機器の状態を参照する場合、「Current *形容詞 VariableName*」とする。"*VariableName*"は、該当する VariableName

が入る。*形容詞*には、分割している項目を示す用語を格納する。例えば2バイトごとに冷房、暖房、除湿、送風モード時それぞれの定格消費電力を定義するECHONETプロパティを制御する場合、各*形容詞*はそれぞれ、Cool, Heat, Dehumid, Blast となり、各Argument は 、 NewCoolRatedConsumpPower, NewHeatRatedConsumpPower, NewDehumidRatedConsumpPower, NewBlastRatedConsumpPower,となる。

３．４ プロパティの分類と命名ルールのまとめ

プロパティの分類と命名ルールについて、３．２及び３．３で記述したが、プロパティタイプ、型分類、目的語、Action 接頭語、dataType の関係を表 ３．５にまとめる。

表 ３．５ プロパティ分類と命名ルール

プロパティタイプ	型分類	目的語	Action 接頭語	dataType
数値型	AVR / Value 型	数値の種別	Write / Read	ui1, ui2, ui4, i1, i2, i4, float
日付型	Date 型	Date	Set / Get	Date
時刻型	Date 型	Time	Set / Get	Time
レベル型	AVL 型	Level	Set / Get	String
文字表記型	String 型	Code	Set / Get	String
リセット型	AVL 型	–	Reset	String
切り替え型	AVL 型	Status	Set / Get	String
選択型	AVL 型	Status	Set / Get	String
その他型	その他型	–	Set / Get	bin.hex

第４章 UPnP デバイス提供方式時の ECHONET-UPnP ゲートウェイの処理

ECHONET-UPnP ゲートウェイが UPnP 上のデバイスとして動作する場合の処理を以下に記述する。

- プラグアンドプレイ処理
- UPnP コントロールポイントから ECHONET 機器の制御
- ECHONET 機器の状態を UPnP コントロールポイントへ通知

４．１ プラグアンドプレイ処理

図 ３．１に記述するように、本システムに参入している機器を「UPnP コントロールポイント（AV 機器)」、「ECHONET-UPnP ゲートウェイ」、「ECHONET 機器」と三つに分別できる。プラグアンドプレイ処理に関して、ECHONET-UPnP ゲートウェイがネットワークに接続した場合、ECHONET 機器がネットワークに接続した場合について記述する。また、UPnP コントロールポイントがネットワークに接続した場合については、ECHONET 側での処理はないため、本規格書においては記述しない。

ただし、ECHONET 機器のネットワークからの離脱時の処理については、規格の対象外とする。

４．１．１ ECHONET-UPnP ゲートウェイがネットワークに接続する場合

ECHONET-UPnP ゲートウェイが ECHONET ネットワークに接続する場合の処理について記述する。ECHONET-UPnP ゲートウェイが、ECHONET ネットワークに接続する場合の処理については、ECHONET 規格書第２部５．３を参照すること。

ECHONET-UPnP ゲートウェイは ECHONET ネットワークに接続した後に、UPnP ネットワークに接続する。ECHONET-UPnP ゲートウェイが UPnP ネットワークに接続する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。シーケンスを図 ４．１に記述する。

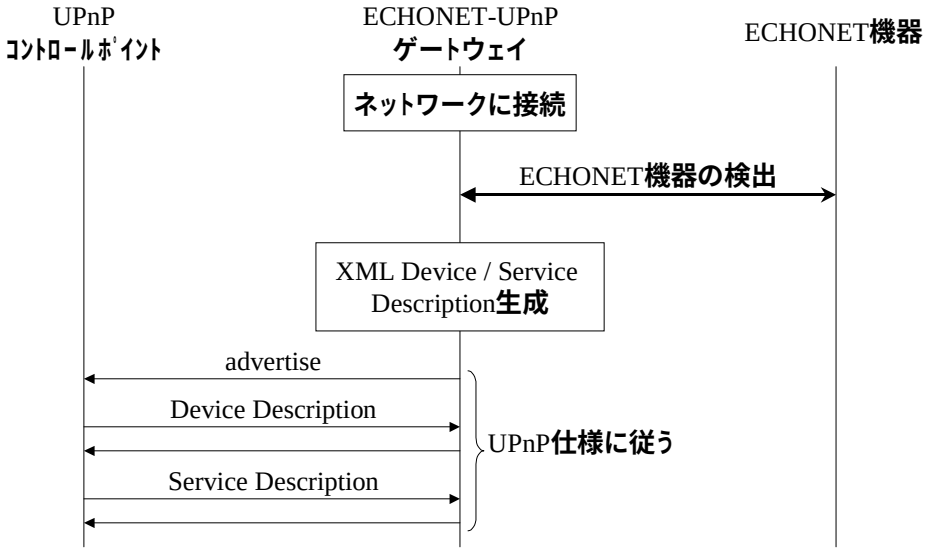

図 ４．１　ECHONET-UPnP ゲートウェイがネットワークに接続する場合

ECHONET-UPnP ゲートウェイは UPnP ネットワークに接続すると、ECHONET ネットワークに接続している ECHONET 機器を検出した結果から XML Device Description 及び XML Service Description を生成する。もしくは、前回起動時に保持している各 ECHONET 機器に対応する XML Device Description 及び XML Service Description を使用する。ECHONET-UPnP ゲートウェイが ECHONET 機器を検知する例を以下に記述する。例えば、ECHONET-UPnP ゲートウェイは起動時に、ECHONET ネットワークにドメイン内一斉同報にて、ノードプロファイルクラス宛に自ノードインスタンスリスト S プロパティ読み出し要求を送信する。ECHONET 機器からその応答を受信することによって、ECHONET 機器の検知を行うことができる。

ECHONET-UPnP ゲートウェイは、ECHONET ネットワークに接続する ECHONET 機器に対応する XML Device Description 及び XML Service Description を保持した後に、advertise をマルチキャストで送信する。advertise の送信以後は、UPnP の仕様に準拠する。

４．１．２ ECHONET 機器がネットワークに接続する場合

ECHONET 機器がネットワークに接続する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。シーケンスを図 ４．２に記述する。

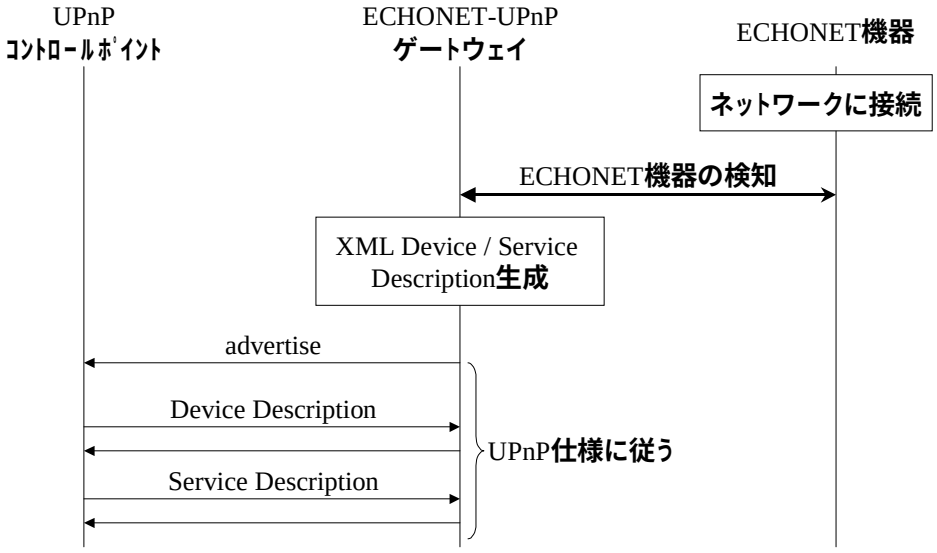

図 ４．２　ECHONET 機器がネットワークに接続する場合

ECHONET-UPnP ゲートウェイは ECHONET 機器がネットワークに接続したことを検知すると、ECHONET 機器に該当する XML Device Description 及び XML Service Description を生成する。もしくは、ECHONET 機器が前回起動していた時に保持していた、ECHONET 機器に対応する XML Device Description 及び XML Service Description を使用する。ECHONET 機器がネットワークに接続したことを ECHONET-UPnP ゲートウェイが検知する例を以下に記述する。例えば、ECHONET 機器が起動時に送信するインスタンス変化クラス通知を ECHONET-UPnP ゲートウェイは受信することによって、ECHONET 機器検知のトリガとすることができる。

ECHONET-UPnP ゲートウェイは、XML Device Description 及び XML Service Description を記述した後に、advertise をマルチキャストで送信する。advertise の送信以後は、UPnP の仕様に準拠する。

４．２ UPnP コントロールポイントから ECHONET 機器の制御

４．２．１ ECHONET 機器の制御

UPnP コントロールポイントが ECHONET-UPnP ゲートウェイを介して、ECHONET 機器を制御する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。

ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信した場合、同期型シーケンスもしくは非同期型シーケンスに従って ECHONET 機器を制御する。

４.２.１.１　同期型による ECHONET 機器の制御

同期型シーケンスに従って ECHONET 機器を制御する場合のシーケンスを図 ４．３に

示す。

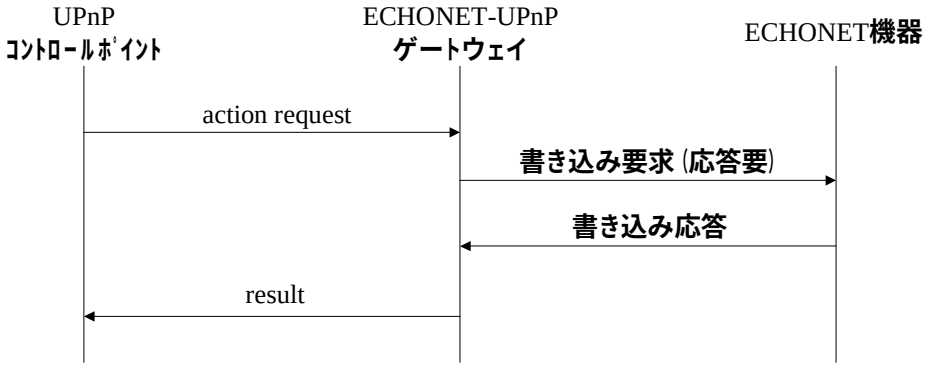

図 ４．３ 同期型シーケンス

同期型シーケンスで ECHONET-UPnP ゲートウェイは動作する場合について、記述する。ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信すると、ECHONET プロトコルのデータフォーマットに変換して、ECHONET 機器へ応答要書き込み要求電文を送信する。その後、ECHONET-UPnP ゲートウェイは、ECHONET 機器から書き込み応答を受信すると、action request 送信元の UPnP コントロールポイントへ result を応答する。

4.2.1.2 非同期型による ECHONET 機器の制御

非同期型シーケンスに従って ECHONET 機器を制御する場合のシーケンスを図 ４．４ に示す。

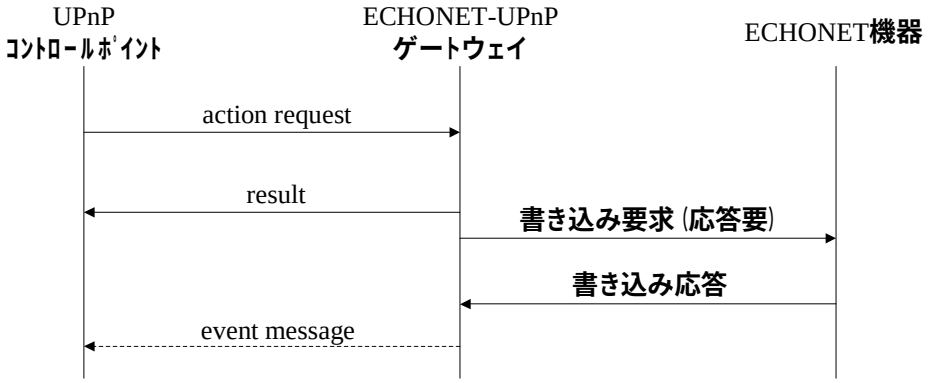

図 ４．４ 非同期型シーケンス

非同期型シーケンスで ECHONET-UPnP ゲートウェイは動作する場合について、記述する。ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信すると、action request 送信元の UPnP コントロールポイントへ result を応答する。それと同様のタイミングで ECHONET プロトコルのデータフォーマットに変換して、ECHONET 機器へ応答要書き込み要求電文を送信する。その後、ECHONET-UPnP ゲートウェイは、ECHONET 機器から書き込み応答を受信すると、subscription request を受

付けていた状態である場合、制御内容を event message として UPnP コントロールポイントへ通知する。

４．２．２ ECHONET 機器の状態参照

UPnP コントロールポイントが ECHONET-UPnP ゲートウェイを介して、ECHONET 機器の状態を参照する場合のシーケンス及び ECHOENT-UPnP ゲートウェイの処理について記述する。

ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信した場合、同期型シーケンスもしくは非同期型シーケンスに従って ECHONET 機器の状態を参照する。

4.2.2.1 同期型による ECHONET 機器の状態参照

同期型シーケンスに従って ECHONET 機器の状態を参照する場合のシーケンスを図４．５に示す。

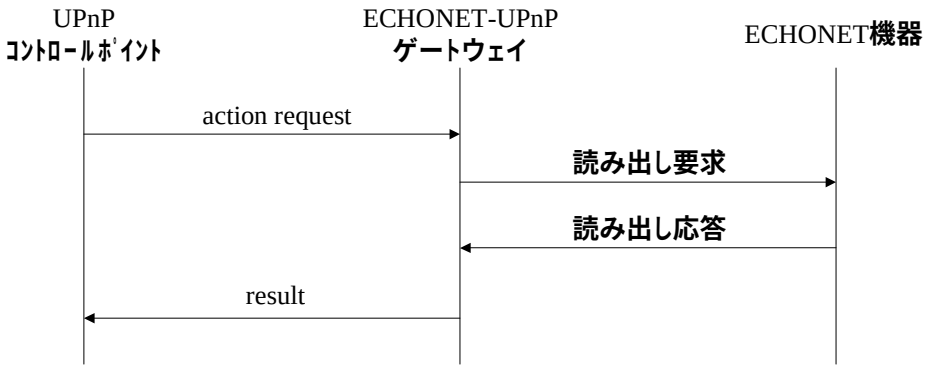

図 ４．５ 同期型シーケンス

同期型シーケンスで ECHONET-UPnP ゲートウェイは動作する場合について、記述する。ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信すると、ECHONET プロトコルのデータフォーマットに変換して、ECHONET 機器へ応答要読み出し要求電文を送信する。その後、ECHONET-UPnP ゲートウェイは、ECHONET 機器から読み出し応答を受信すると、action request 送信元の UPnP コントロールポイントへ result を応答する。

4.2.2.2 非同期型による ECHONET 機器の状態参照

非同期型シーケンスに従って ECHONET 機器の状態を参照する場合のシーケンスを図４．６に示す。

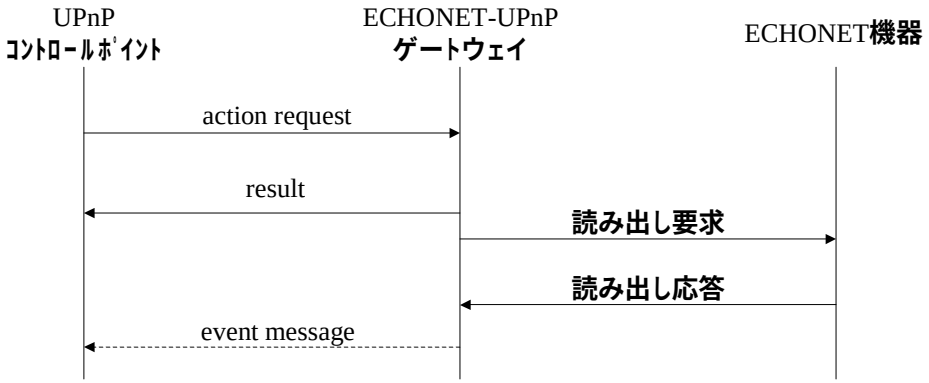

図 ４．６ 非同期型シーケンス

非同期型シーケンスで ECHONET-UPnP ゲートウェイは動作する場合について、記述する。ECHONET-UPnP ゲートウェイは UPnP コントロールポイントから action request を受信すると、action request 送信元の UPnP コントロールポイントへ result を応答する。それと同様のタイミングで ECHONET プロトコルのデータフォーマットに変換して、ECHONET 機器へ応答要読み出し要求電文を送信する。その後、ECHONET-UPnP ゲートウェイは、ECHONET 機器から読み出し応答を受信すると、subscription request を受付けていた場合、応答に含まれる機器の状態を event message として UPnP コントロールポイントへ通知する。

また、ECHONET-UPnP ゲートウェイは非同期型シーケンスで動作する場合、ECHONET 機器の状態を保持しておかなければならない。なぜなら、該当する機器の状態を action request 受信直後の result 応答に含める必要があるためである。

４．３ ECHONET 機器の状態通知

ECHONET-UPnP ゲートウェイが ECHONET 機器の状態を UPnP コントロールポイントへ通知する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。

event message を通知するタイミングは特には規定しないが、以下の場合に event message を通知することを推奨とする。

- ECHONET 機器から状態変化通知を受信した場合
- ECHONET-UPnP ゲートウェイが ECHONET 機器の状態を取得し、ECHONET 機器の状態が変更したことを検知した場合
- ECHONET-UPnP ゲートウェイが非同期型シーケンスで動作中に、ECHONET 機器から応答を受信した場合（図 ４．４、図 ４．６参照）

ECHONET-UPnP ゲートウェイが、ECHONET 機器から状態変化通知を受信し、UPnP コントロールポイントへ event message を送信する場合のシーケンスを図 ４．７に示す。

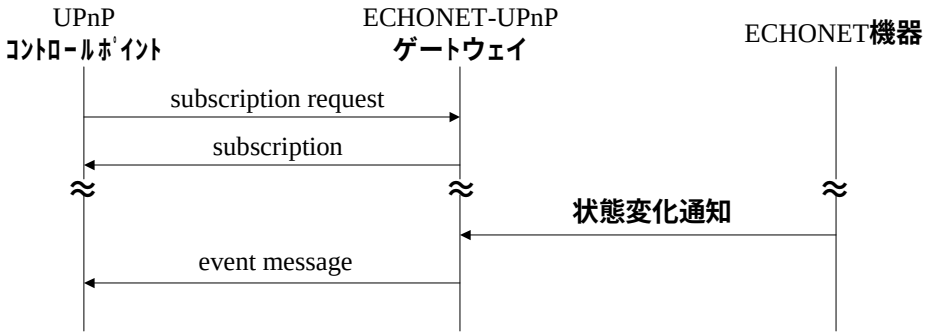

図 ４．７ イベント通知を行う場合

ECHONET-UPnP ゲートウェイは、subscription request を登録している UPnP コントロールポイントにユニキャストで event message を送信する。

第５章 Device Template

ECHONET 機器の情報を UPnP 側へ公開するための設計ポリシーとして、ECHONET 機器オブジェクトごとに、対応する UPnP 仮想デバイスを UPnP ルートデバイスとして、ECHONET 機器の情報を UPnP へ公開することとする。各 UPnP 仮想デバイスはそれぞれ ECHONET Service を提供するものとする。

５．１ Device の定義

５．１．１ Device Type

下記デバイスタイプは、この Device Template に準拠したデバイスである。

urn:echonet-gr-jp:device:ECHONET_***Appliance***:1

ECHONET 特有の要素として、XML Device Description 内に記述するドメイン名は、”echonet-gr-jp”とする。

「:1」は仕様のバージョン情報を示している。”***Appliance***”の部分には、Appendix ２に規定する Appliance 名称で置き換える。例えば、家庭用エアコンクラスの場合、Appliance 名称は”**HomeAirConditioner**”であり、これをあてはめることで Device Type は

”urn:echonet-gr-jp:device:ECHONET_HomeAirConditioner:1”と記述することとする。

５．１．２ Device の要件

Device の Type が urn:echonet-gr-jp:device:ECHONET_***Appliance***:1 である機器は、以下に示す規定を満たす機能を備える必要がある。表 ５．１に Device Requirements を記載する。

表 ５．１ Device Requirements

DeviceType	Root	Req. or Opt.[1]	ServiceType	Req. or Opt.[1]	Service ID[2]
ECHONET_***Appliance***:1	Root	R	ECHONET_Service:1	R	ECHONET_***Appliance***

[1] R=Required, O=Optional, X=Non-standard.
[2] Prefixed by urn:echonet-gr-jp:serviceId: .

ECHONET_Service の Service ID を urn:echonet-gr-jp:serviceId:ECHONET_Appliance と規定する。Appliance は、ECHONET 機器固有の名称に置き換えることとし、具体的な名称については、ECHONET 規格書 Appendix２に規定する。

例えば、エアコンの場合、上記”Appliance”の部分は”HomeAirConditioner”であり、こ

れをあてはめることで Service ID は

“urn:echonet-gr-jp:serviceId:ECHONET_HomeAirConditioner”

と記述することとする。

５．２ XML Device Description

表 ５．２に、５．１に記載した Device の XML Device Description のフォーマットを記述する。なお、XML Device Description の各要素の仕様については、UPnP Device Architecture Ver.1.0 に準拠するものとする。

表 ５．２　XML Device Description のフォーマット

```
<?xml version="1.0"?>
<root xmlns="urn:echonet-gr-jp:device-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <URLBase>base URL for all relative URLs</URLBase>
    <device>
        <deviceType>urn:echonet-gr-jp:device:ECHONET_Appliance:1</deviceType>
        <friendlyName>short user-friendly title</friendlyName>
        <manufacturer>manufacturer name</manufacturer>
        <manufacturerURL>URL to manufacturer site</manufacturerURL>
        <modelDescription>long user-friendly title</modelDescription>
        <modelName>model name</modelName>
        <modelNumber>model number</modelNumber>
        <modelURL>URL to model site</modelURL>
        <serialNumber>manufacturer's serial number</serialNumber>
        <UDN>uuid:UUID</UDN>
        <UPC>Universal Product Code</UPC>
        <iconList>
            <icon>
                <mimetype>image/format</mimetype>
                <width>horizontal pixels</width>
                <height>vertical pixels</height>
                <depth>color depth</depth>
                <url>URL to icon</url>
            </icon>
            XML to declare other icons, if any, go here
        </iconList>
        <serviceList>
            <service>
                <serviceType>urn:echonet-gr-jp:service:ECHONET_Service:1</serviceType>
                <serviceId>urn:echonet-gr-jp:serviceId:ECHONET_Appliance</serviceId>
                <SCPDURL>URL to service description</SCPDURL>
                <controlURL>URL for control</controlURL>
```

```
            <eventSubURL>URL for eventing</eventSubURL>
        </service>
</serviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

要素名「UDN」については、ECHONET-UPnP ゲートウェイが、ECHONET 機器のネットワークへの接続を検知した場合に、UUID を ECHONET 機器ごとに固有となるよう UUID を生成し、UDN 要素に格納し、XML Device Description に記述するものとする。

５．３ エアコンの XML Device Description の例

参考例として、エアコン(EOJ = 0x013000)に対する UPnP 仮想デバイスの XML Device Description の例を以下に示す。

表 ５．３ XML Device Description の実例（エアコン）

```
<root xmlns="urn:echonet-gr-jp:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:echonet-gr-jp:device:ECHONET_HomeAirConditioner:1</deviceType>
    <friendlyName>Home Air Conditioner</friendlyName>
    <manufacturer>manufacturer name of air conditioner</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>Home Air Conditioner</modelDescription>
    <modelName>model name of air conditioner</modelName>
    <UDN>uuid:ad82f4cd-bafd-11da-9d2c-000e7b032792</UDN>
    <serviceList>
      <service>
        <serviceType>urn:echonet-gr-jp:service:ECHONET_Service:1</serviceType>
        <serviceId>urn:echonet-gr-jp:serviceId:ECHONET_HomeAirConditioner</serviceId>
        <SCPDURL>/service.xml</SCPDURL>
        <controlURL>/ECHONET/control/ECHONET_HomeAirConditioner1</controlURL>
        <eventSubURL>/ECHONET/Eventing/ECHONET_HomeAirConditioner1</eventSubURL>
      </service>
    </serviceList>
    <presentationURL>/presentation.html</presentationURL>
```

```
    </device>
</root>
```

第６章 Service Template

本プロジェクトにおいて、ECHONET_*Appliance* が保持する ServiceType である ECHONET_Service の Service Template を下記に記載する。

６．１ サービスモデルの定義

６．１．１ Service Type

以下に示す Service Type を持つサービスは本サービステンプレートに準拠する。
urn:echonet-gr-jp:service:ECHONET_Service:1
ECHONET 特有の要素として、XML Device Description 内に記述するドメイン名は、”echonet-gr-jp”とする。

６．１．２ Service Type の要件

サービスタイプ ECHONET_Service は、以下の規定を満たす必要がある。
Variable Name には、３．３に記述した命名ルールに基づいた名称を記述する。Data Type、Allowed Value、Eng.Units の具体的な仕様に関しては、ECHONET 規格書 Appendix 及び Appendix ２を参照すること。

表 ６．１ State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value	Default Value	Eng. Units
VariableName	see ECHONET Specification	see ECHONET Specification	see ECHONET Specification	*None*	see ECHONET Specification

[1] R=Required, O=Optional, X=Non-standard.

・ Variable Name
”*VariableName*”に入力する具体的な文字列は、ECHONET 規格書 Appendix ２を参照すること。例えば、動作状態（EPC=0x80）を記述する場合、VariableName は、”OperationStatus”と記述することになる。
・ Req. or Opt.
必須かオプションかは ECHONET 規格書 Appendix を参照すること。
・ Data Type
ECHONET 規格書 Appendix ２を参照すること。
・ Allowed Value
ECHONET 規格書 Appendix２を参照すること。

・ Default Value
デフォルト値は、すべての ECHONET プロパティに対して規定しない。

・ Eng. Units
ECHONET 規格書 Appendix を参照すること。

６．１．３ Action

VariableName に記載したプロパティに対して、状態参照を行えるサービスか、及び制御を行えるサービスかということを規定する。

表 ６．２ Action の記述

Name	Req. or Opt.
Set ***VariableName***	Opt.
Get ***VariableName***	Opt.
Write ***VariableName***	Opt.
Read ***VariableName***	Opt.
Reset ***VariableName***	Opt.

・ Name
制御を行う場合、Set、Write を Variable Name の接頭語としたものを Action 名とする。状態参照を行う場合、Get、Read を Variable Name の接頭語としたものを Action 命名規則とする。また、状態をリセットする場合、Reset を Variable Name の接頭語としたものを Action 名とする。使用する接頭語の規則については、表 ３．３を参照すること。

・ Req. or Opt.
Optional とする。

○ Set ***VariableName***、Write ***VariableName***
ECHONET プロパティの状態を変更する場合に使用する。

－ 引数
アクションが Set ***VariableName***、Write ***VariableName*** である場合の Arguments を表６．３に記載する。

表 ６．３ 制御時の引数リスト

Argument	Direction	relatedStateVariable
New ***VariableName***	In	***VariableName***

＋ Argument
引数を示す。New を Variable Name の接頭語として、New ***VariableName*** を Argument 名とする。ただし、複合型の ECHONET プロパティである場合、New ***形容詞 VariableName*** となる。動作状態を例にすると、Argument は NewOperationStatus

となる。

＋ Direction

Direction は機器を制御する場合は、In となる。

＋ relatedStateVariable

制御後の反映先を示す。”***VariableName***” を用いるものとする。ただし、複合型の ECHONET プロパティである場合、”***形容詞 VariableName***”を用いるものとする。

○ Get***VariableName***、Read***VariableName***

ECHONET プロパティの状態を参照する場合に使用する。

－ 引数

アクションが Get***VariableName***、Read***VariableName*** である場合の Arguments を表６-４に記載する。

表 ６-４ 状態参照時の引数リスト

Argument	Direction	relatedStateVariable
Current***VariableName***	Out	***VariableName***

＋ Argument

引数を示す。Current を ***VariableName*** の接頭語として、Current***VariableName*** を Argument 名とする。ただし、複合型の ECHONET プロパティである場合、Current ***形容詞 VariableName*** となる。動作状態を例にすると、Argument は CurrentOperationStatus となる。

＋ Direction

Direction は機器の状態を参照する場合は、Out となる。

＋ relatedStateVariable

状態参照先を示す。***VariableName*** を用いるものとする。ただし、複合型の ECHONET プロパティである場合、”***形容詞 VariableName***”を用いるものとする。

○ Reset***VariableName***

ECHONET プロパティの状態をリセットする場合に使用する。Argument は存在しない。

６．２ XML Service Description

表 ６．５に、ECHONET_Service の XML Service Description のフォーマットを記述する。プロパティタイプや分類型によって、使用する XML Service Description のフォーマットを決定することができる。なお、各要素の仕様については、UPnP Device Architecture Ver.1.0 に準拠するものとする。

表 ６．５ XML Service Description のフォーマット

```
<?xml version="1.0"?>
<scpd xmlns="urn:echonet-gr-jp:service-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <actionList>
        <!-- 日付型、時刻型、レベル型、文字表記型、選択型、切り替え型、その他型 -->
        <action>
            <name>Set VariableName</name>
            <argumentList>
                <argument>
                    <name>New VariableName</name>
                    <direction>in</direction>
                    <relatedStateVariable>VariableName</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <!-- 日付型、時刻型、レベル型、文字表記型、選択型、切り替え型、その他型 -->
        <action>
            <name>Get VariableName</name>
            <argumentList>
                <argument>
                    <name>Current VariableName</name>
                    <direction>out</direction>
                    <relatedStateVariable>VariableName</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <!-- 数値型-->
        <action>
            <name>Write VariableName</name>
            <argumentList>
                <argument>
                    <name>New VariableName</name>
                    <direction>in</direction>
                    <relatedStateVariable>VariableName</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <!-- 数値型 -->
        <action>
            <name>Read VariableName</name>
            <argumentList>
                <argument>
                    <name>Current VariableName</name>
```

```
                <direction>out</direction>
                <relatedStateVariable>VariableName</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <!-- リセット型 -->
    <action>
        <name>ResetVariableName</name>
    </action>
    Declarations for other actions added by UPnP vendor (if any) go here
</actionList>
<serviceStateTable>
    <!-- AVR型 -->
    <stateVariable sendEvents="yes" or "no">
        <name>VariableName</name>
        <dataType>see ECHONET Specifications</dataType>
        <defaultValue>defaultValue</defaultValue>
        <allowedValueRange>
            <minimum>Minimum Data Value</minimum>
            <maximum>Maximum Data Value</maximum>
            <step>Step Value</step>
        </allowedValueList>
    </stateVariable>
    <!-- Value型 -->
    <stateVariable sendEvents="yes" or "no">
        <name>VariableName</name>
        <dataType>see ECHONET Specifications</dataType>
        <defaultValue>defaultValue</defaultValue>
    </stateVariable>
    <!-- Date型 -->
    <stateVariable sendEvents="yes" or "no">
        <name>VariableName</name>
        <dataType>see ECHONET Specifications</dataType>
        <defaultValue>defaultValue</defaultValue>
    </stateVariable>
    <!-- AVL型 -->
    <stateVariable sendEvents="yes" or "no">
        <name>VariableName</name>
        <dataType>string</dataType>
        <defaultValue>defaultValue</defaultValue>
        <allowedValueList>
            <allowedValue>Property Data1</allowedValue>
            Declarations for other allowed values added by UPnP vendor (if any) go here
        </allowedValueList>
    </stateVariable>
    <!-- String型 -->
```

```
        <stateVariable sendEvents="yes" or "no">
            <name>VariableName</name>
            <dataType>string</dataType>
            <defaultValue>defaultValue</defaultValue>
        </stateVariable>
        <!-- その他型 -->
        <stateVariable sendEvents="yes" or "no">
            <name>VariableName</name>
            <dataType>bin.hex</dataType>
            <defaultValue>defaultValue</defaultValue>
        </stateVariable>
        Declarations for other state variables added by UPnP vendor (if any) go here
    </serviceStateTable>
</scpd>
```

６．３ エアコンの XML Service Description の例

参考例として、エアコン(EOJ = 0x013000)に対する UPnP 仮想デバイスの XML Service Description の例を以下に示す。

表 ６．６ XML Device Description の実例（エアコン）

```
<scpd xmlns="urn:echonet-gr-jp:service-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <actionList>
        <action>
            <name>SetOperationStatus</name>
            <argumentList>
                <argument>
                    <name>NewOperationStatus</name>
                    <direction>in</direction>
                    <relatedStateVariable>OperationStatus</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>GetOperationStatus</name>
            <argumentList>
                <argument>
```

```
                <name>CurrentOperationStatus</name>
                <direction>out</direction>
                <relatedStateVariable>OperationStatus</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetProductCode</name>
        <argumentList>
            <argument>
                <name>CurrentProductCode</name>
                <direction>out</direction>
                <relatedStateVariable>ProductCode</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>SetOperationModeStatus</name>
        <argumentList>
            <argument>
                <name>NewOperationModeStatus</name>
                <direction>in</direction>
                <relatedStateVariable>OperationModeStatus</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>GetOperationModeStatus</name>
        <argumentList>
            <argument>
                <name>CurrentOperationModeStatus</name>
                <direction>out</direction>
                <relatedStateVariable>OperationModeStatus</relatedStateVariable>
            </argument>
        </argumentList>
    </action>
    <action>
        <name>WriteDesiredTemp</name>
        <argumentList>
            <argument>
                <name>NewDesiredTemp</name>
```

```
                    <direction>in</direction>
                    <relatedStateVariable>DesiredTemp</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>ReadDesiredTemp</name>
            <argumentList>
                <argument>
                    <name>CurrentDesiredTemp</name>
                    <direction>out</direction>
                    <relatedStateVariable>DesiredTemp</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>SetWindVolumeLevel</name>
            <argumentList>
                <argument>
                    <name>NewWindVolumeLevel</name>
                    <direction>in</direction>
                    <relatedStateVariable>WindVolumeLevel</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>GetWindVolumeLevel</name>
            <argumentList>
                <argument>
                    <name>CurrentWindVolumeLevel</name>
                    <direction>out</direction>
                    <relatedStateVariable>WindVolumeLevel</relatedStateVariable>
                </argument>
            </argumentList>
        </action>
    </actionList>
    <serviceStateTable>
        <stateVariable sendEvents="yes">
            <name>OperationStatus</name>
            <dataType>string</dataType>
            <allowedValueList>
```

```
                <allowedValue>ON</allowedValue>
                <allowedValue>OFF</allowedValue>
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>ProductCode</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>OperationModeStatus</name>
            <dataType>string</dataType>
            <allowedValueList>
                <allowedValue>Auto</allowedValue>
                <allowedValue>Cooling</allowedValue>
                <allowedValue>Heating</allowedValue>
                <allowedValue>Dehumidifying</allowedValue>
                <allowedValue>Blast</allowedValue>
                <allowedValue>Other</allowedValue>
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>DesiredTemp</name>
            <dataType>ui1</dataType>
            <allowedValueRange>
                <minimum>16</minimum>
                <maximum>30</maximum>
                <step>1</step>
            </allowedValueRange>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>WindVolumeLevel</name>
            <dataType>string</dataType>
            <allowedValueList>
                <allowedValue>1</allowedValue>
                <allowedValue>2</allowedValue>
                <allowedValue>3</allowedValue>
                <allowedValue>4</allowedValue>
                <allowedValue>5</allowedValue>
                <allowedValue>6</allowedValue>
                <allowedValue>7</allowedValue>
                <allowedValue>8</allowedValue>
```

```
                    <allowedValue>Auto</allowedValue>
                </allowedValueList>
            </stateVariable>
        </serviceStateTable>
</scpd>
```

第７章 ECHONET オブジェクト提供方式

本章では ECHONET オブジェクト提供方式として ECHONET-UPnP ゲートウェイが動作する場合について記述する。

７．１ 基本的な考え方

本 ECHONET オブジェクト提供方式はオプションである。また、実装の際は UPnP 提供方式を必ず搭載することとする。

ECHONET オブジェクト提供方式の ECHONET-UPnP ゲートウェイは、UPnP デバイスの提供するサービスを ECHONET ネットワーク上に、仮想的に ECHONET オブジェクトとして提供する。以下、その仮想的な ECHONET オブジェクトを、ECHONET オブジェクトと呼称する。ECHONET ネットワークにおいて、UPnP デバイスに対応する ECHONET オブジェクトは、ECHONET 規格に従って動作するものとし、従来の ECHONET オブジェクトとして同様に扱うことができるものとする。

図 ７．１に ECHONET オブジェクト提供方式部分のシステム構成図を示す。

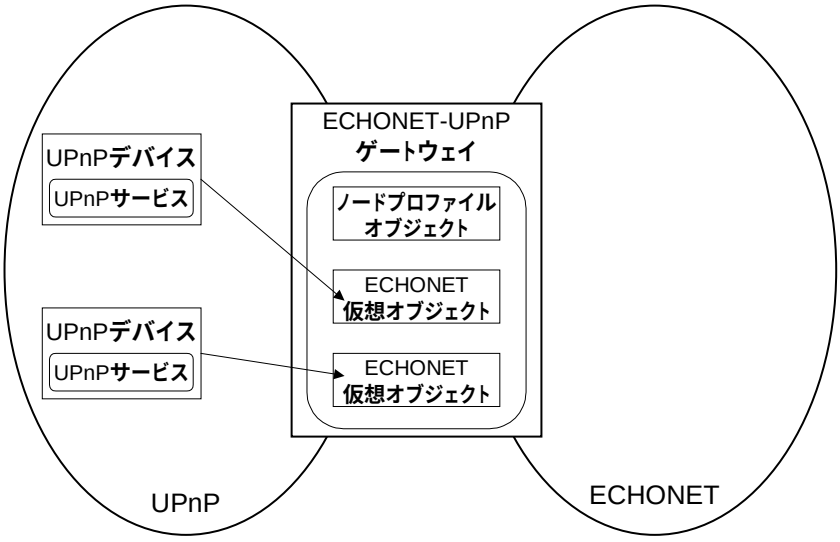

図 ７．１ ECHONET オブジェクト提供方式時のシステム構成例

第８章 ECHONET オブジェクト提供方式時の ECHONET-UPnP ゲートウェイの処理

本章では、ECHONET-UPnP ゲートウェイが ECHONET 上の ECHONET オブジェクトとして動作する場合の処理を以下に記述する。

・ プラグアンドプレイ処理
・ ECHONET オブジェクトから UPnP デバイスの制御

８．１ プラグアンドプレイ処理

プラグアンドプレイ処理に関して、ECHONET-UPnP ゲートウェイが接続した場合と、UPnP デバイスがネットワークに接続した場合について説明する。コントローラとなる ECHONET オブジェクトがネットワークに接続した場合については、ECHONET-UPnP ゲートウェイの動作に影響を与えるものではないため記述しない。
UPnP デバイスのネットワーク離脱時の ECHONET-UPnP ゲートウェイの処理については、規格の対象外とする。

８．１．１ ECHONET-UPnP ゲートウェイがネットワークに接続する場合

ECHONET-UPnP ゲートウェイがネットワークに接続する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。シーケンスを図 ８．１に示す。

ECHONET-UPnP ゲートウェイは起動したら、UPnP デバイスの存在を把握するために、UPnP ネットワークに対して search メッセージをマルチキャストで送出する。Search に対する response メッセージを受信したら、XML device description 及び XML service description を取得する。取得した XML description の記述に応じて、対応する ECHONET 機器オブジェクトを生成する。生成され増えた ECHONET オブジェクトに対応し、ノードプロファイルオブジェクトのプロパティを修正し、状態変化通知を送出する。送出する状態変化通知は ECHONET 規格に従うものとする。

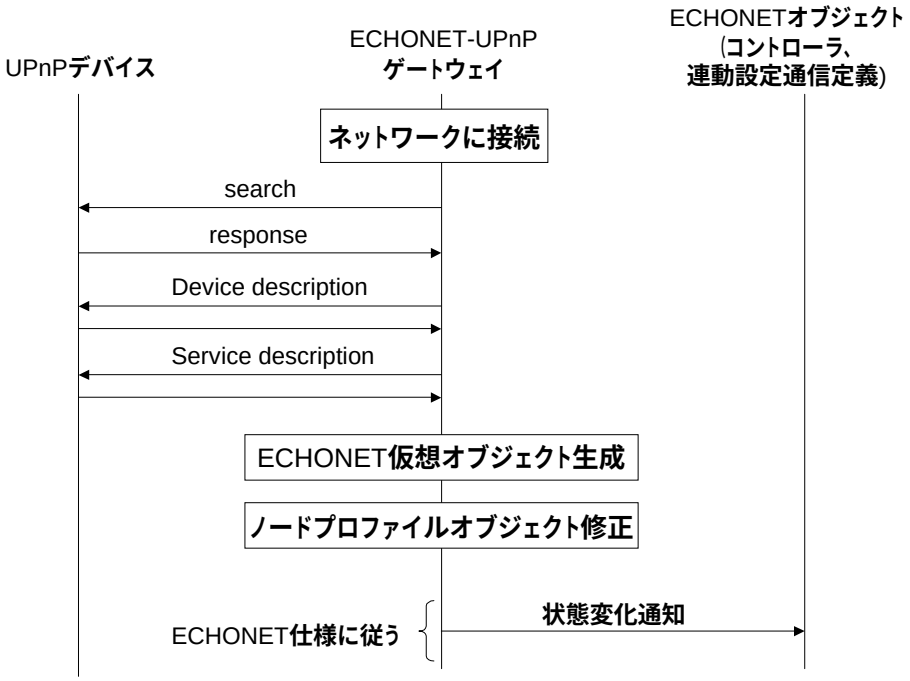

図 ８．１ ECHONET-UPnP ゲートウェイがネットワークに接続する場合

８．１．２ UPnP デバイスがネットワークに接続する場合

UPnP デバイスがネットワークに接続する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。シーケンスを図 ８．２に示す。

ECHONET-UPnP ゲートウェイは、UPnP デバイスがネットワーク接続時にマルチキャストで送出する advertise メッセージを受信したら、XML device description 及び XML service description を取得する。XML description の記述に応じて、対応する ECHONET オブジェクトを生成する。生成され増えた ECHONET オブジェクトに対応し、ノードプロファイルオブジェクトのプロパティを修正し、状態変化通知を送出する。送出する状態変化通知は ECHONET 規格に従うものとする。

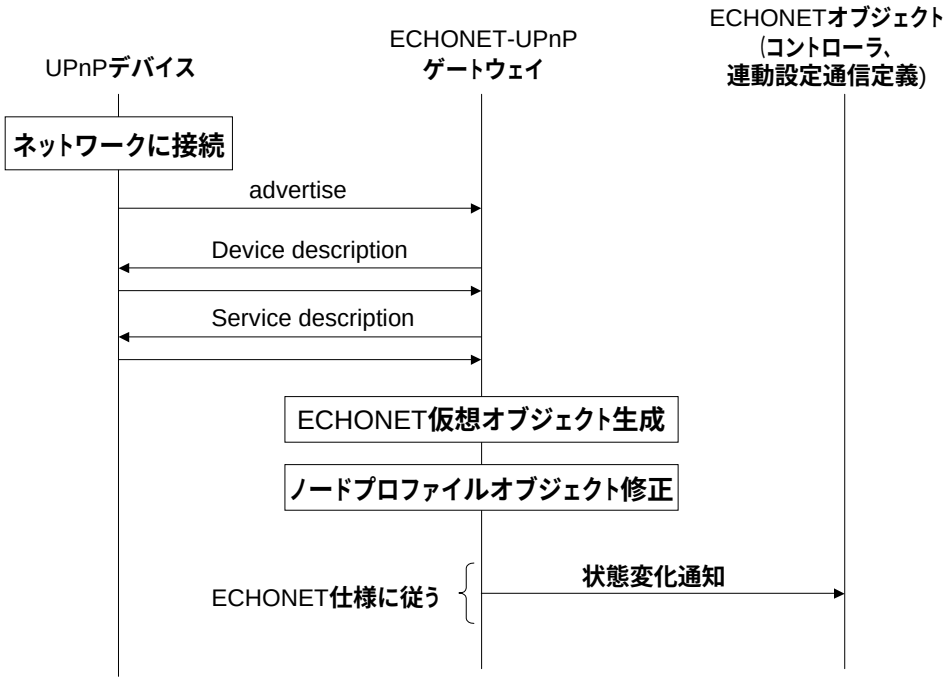

図 ８．２ UPnP デバイスがネットワークに接続する場合

８．２ ECHONET オブジェクトから UPnP デバイスの制御

８．２．１ UPnP デバイスの制御

ECHONET オブジェクト（コントローラオブジェクト、連動設定通信定義オブジェクト etc）が ECHONET-UPnP ゲートウェイを介して、UPnP デバイスを制御する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。ECHONET-UPnP ゲートウェイの動作シーケンスを図 ８．３に示す。

ECHONET-UPnP ゲートウェイは ECHONET オブジェクトから制御要求を受信すると、UPnP プロトコルのデータフォーマットに変換して、UPnP デバイスへ action request を送信する。その後、ECHONET-UPnP ゲートウェイは、UPnP デバイスから result を受信する。ECHONET オブジェクトからの制御要求が、応答要書き込み電文であった場合のみ、制御応答を制御要求送信元の ECHONET オブジェクトに送信する。

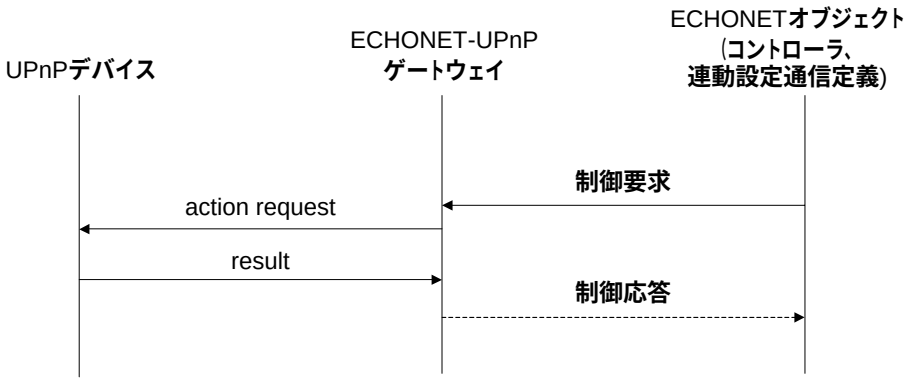

図 ８．３ UPnP デバイス制御シーケンス

８．２．２ UPnP デバイスの状態参照

ECHONET オブジェクト（コントローラオブジェクト、連動設定通信定義オブジェクト etc）が ECHONET-UPnP ゲートウェイを介して、UPnP デバイスの状態を参照する場合のシーケンス及び ECHONET-UPnP ゲートウェイの処理について記述する。ECHONET-UPnP ゲートウェイの動作シーケンスを図 ８．４に示す。

ECHONET-UPnP ゲートウェイは ECHONET オブジェクトから状態参照要求を受信すると、UPnP プロトコルのデータフォーマットに変換して、UPnP デバイスへ action request を送信する。その後、ECHONET-UPnP ゲートウェイは、UPnP デバイスから result を受信する。ECHONET-UPnP ゲートウェイは状態参照応答を状態参照要求送信元の ECHONET オブジェクトに送信する。

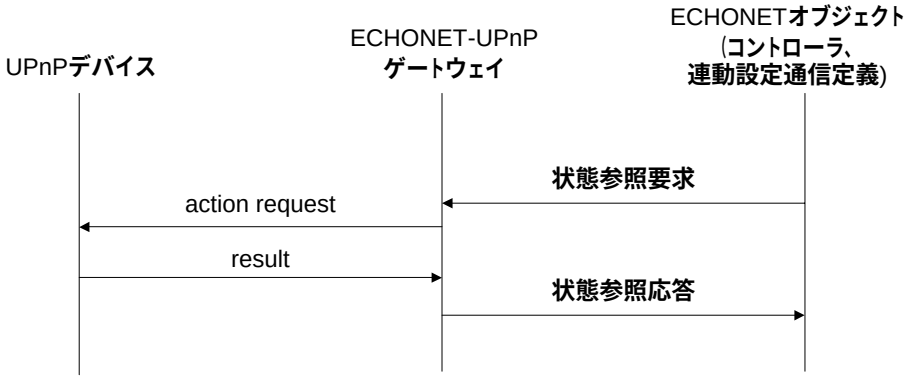

図 ８．４ UPnP デバイス状態参照シーケンス