詳説 iSCP 1.0

第 3 版（2022 年 11 月）

aptpod

Rev. 79807be

目次

01 intdash と iSCP 4

1.1 iSCP のプロトコルスタック 5
1.2 iSCP で定義されている内容 5
1.3 このドキュメントの構成 6

02 intdash のデータモデル 7

2.1 時系列データとデータポイント（時刻を付与されたペイロード） 7
2.2 エッジ（時系列データを送信／受信するデバイス） 8
2.3 計測（ある基準時刻のもとに管理される時系列データポイントの集合） 9
2.4 基準時刻と経過時間 10
2.5 時系列データポイントのデータ構造 11

03 iSCP のメッセージングモデル 14

3.1 アップストリームとダウンストリーム 14
3.2 データの転送 14
3.3 宛先によるデータのフィルタリング 15
3.4 データ識別子によるフィルタリング（ダウンストリーム） 16
3.5 再送フラグと永続化フラグ（アップストリーム） 17

04 ペイロードのパース処理 18

4.1 サーバーサイドにおけるパース処理 18
4.2 Visual M2M Data Visualizer におけるパース処理 21

05 付録: データタイプとペイロードの定義 23

5.1 Bytes（文字列 ID を持ったバイト列データ） 23
5.2 String（文字列 ID を持った文字列データ） 25
5.3 Float（文字列 ID を持った倍精度浮動小数点数） 26
5.4 Int（文字列 ID を持った 64bit 符号付き整数） 27
5.5 JPEG（JPEG 画像） 28
5.6 H.264（H.264/AVC 動画） 28
5.7 PCM（WAVE 形式の音声波形情報） 30
5.8 AAC（ADTS 形式の AAC 音声） 31
5.9 CAN（Controller Area Network 用の通信プロトコル） 32
5.10 MAVLink 1 Packet（Micro Air Vehicle/ドローン用の通信プロトコル） 33
5.11 NMEA（GPS データの通信用プロトコル） 34
5.12 General Sensor（アプトポッド独自定義） 36
5.13 Controlpad（アプトポッド独自定義） 42
5.14 Generic（アプトポッド独自定義） 43
5.15 Base Time - 基準時刻 44

06 付録: 受信データのフィルタリング 46

6.1 Bytes、String、Float、Int のデータに対するフィルター 47

6.2 CAN、General Sensor、NMEA、Control Pad、Generic のデータに対するフィルター 48

01　intdash と iSCP

intdash（INTeractive DAta Streaming Hub）は、高頻度時系列データをストリーミングするデータ転送ブローカーと、ブローカーを介して転送された時系列データを保存・管理するデータストアから成る、産業用 IoT プラットフォームです。

このドキュメントで解説する iSCP（intdash Stream Control Protocol）は、intdash に接続するクライアントがリアルタイムに時系列データを送受信する際に使用する、アプリケーション層の通信プロトコルです。iSCP はアプトポッドが独自に開発したものです。

> **注釈:** iSCP が使用されるのは、リアルタイムデータの送受信のみです。過去データを intdash サーバーから取得する際は、REST API を使って JSON データとして取得しますので、iSCP は使われません。

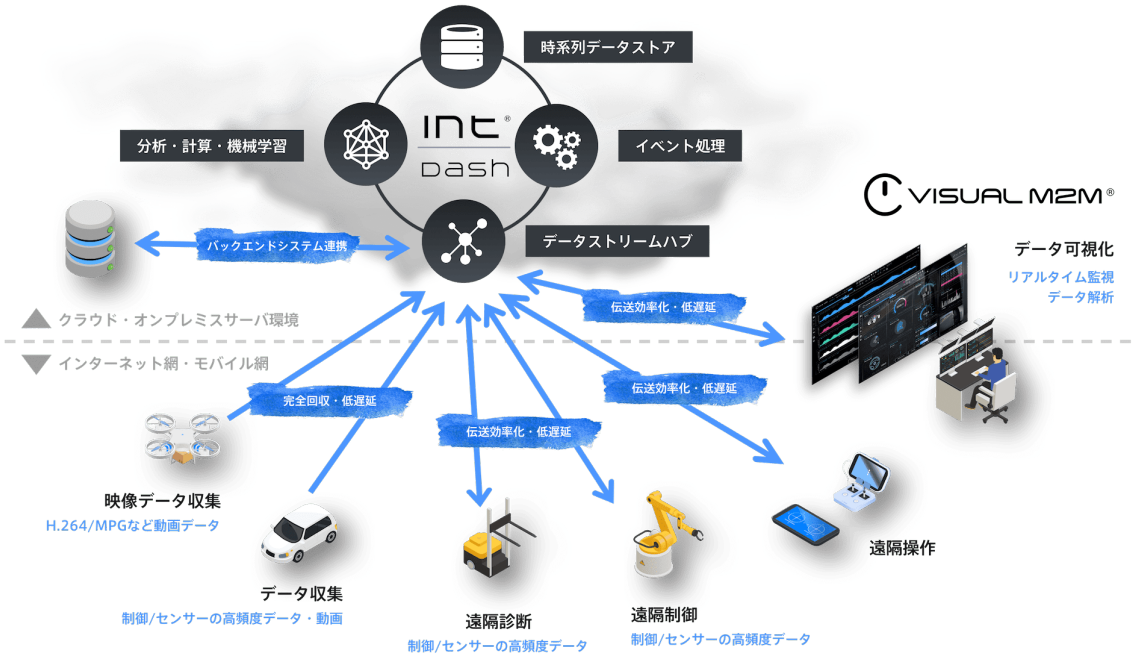


図 1　intdash の全体図

1.1 iSCP のプロトコルスタック

iSCP は、WebSocket をトランスポート層として使用します。WebSocket は TCP をベースにしており、サーバーとエッジの間のデータの到達やその順序は iSCP より下位の TCP 層において保証されます。iSCP の層では、サーバーを介したエッジ間での通信の制御や、TCP コネクションを跨いでのデータ伝送欠損の再送制御を行います。また、認証情報の授受や圧縮などの一部の機能は、WebSocket が持っている機能を利用しています。

iSCP	• エッジとエッジの間のデータ転送、多対多通信 • コネクションを跨った伝送欠損管理、再送制御
WebSocket	• HTTPサーバーでの待ち受け • 認証情報の授受 • 圧縮による効率化
TCP/IP	ブローカーとエッジの間の • 到達保証 • 順序保証 • コネクション管理 • 輻輳制御

図 2　プロトコルスタック

1.2 iSCP で定義されている内容

iSCP では、主に以下の 2 つの内容が定義されています。

- メッセージフォーマットと通信シーケンス (p. 5)
- ペイロードのフォーマット (p. 6)

注釈: iSCP は、アプトポッド製品間の通信に使用する内部プロトコルであり、その詳細仕様については現在公開していません。

1.2.1 メッセージフォーマットと通信シーケンス

通信に使用するメッセージの内容やフォーマット、およびそのメッセージの授受シーケンスが定義されています。定義されたシーケンスは、エッジがデータの送信や受信を開始するまでのシーケンス（ネゴシエーション）と、ネゴシエーションが完了した後に開始される時系列データの伝送シーケンスの 2 つに大別されます。

intdash SDK や intdash Edge Agent などのデベロッパー向け開発キットを利用する場合、これらの製品が各メッセージの内容や通信シーケンスの仕様を隠蔽してくれるため、その詳細について理解しておく必要はありません。しかし、ネゴシエーション時のリクエストパラメーターの一部は、ユーザーが指定する必要があります。ネゴシエーション時に必要な情報については、iSCP のメッセージングモデル (p. 14) の中で解説していますので参照してください。

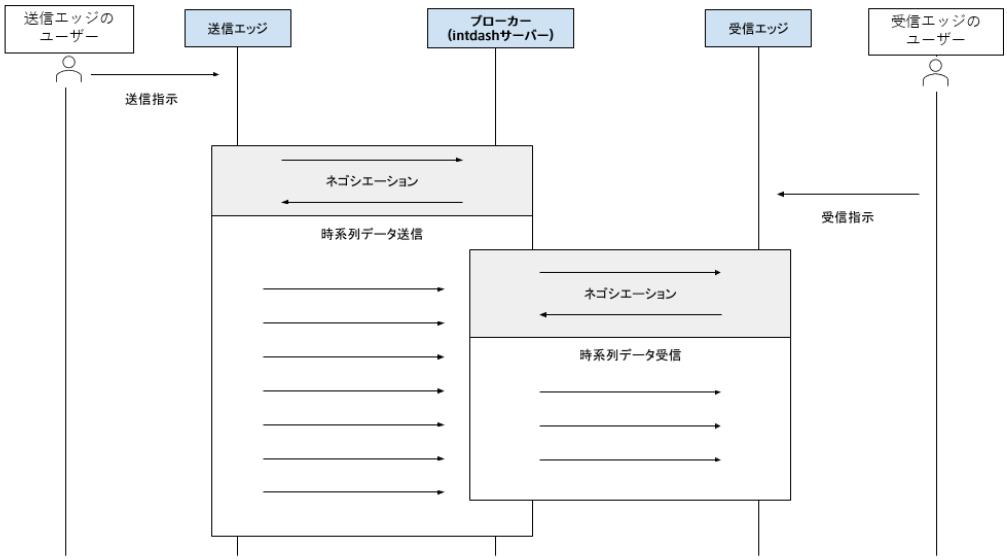


図 3　エッジからエッジにデータを送信する際の通信シーケンスの概略

1.2.2 ペイロードのフォーマット

iSCP では、時系列データを格納するペイロードのフォーマットについても規定しており、データ送信時には、事前に規定されているフォーマットに沿ってデータを格納する必要があります。ペイロードのフォーマットについては、付録: データタイプとペイロードの定義 (p. 23) を参照してください。

1.3 このドキュメントの構成

次章以降では、以下の順で iSCP について説明します。

intdash のデータモデル (p. 7)
iSCP を理解するための前提となる、intdash のデータモデルについて説明します。「計測」、「データポイント」といった、intdash の基本的な概念についても説明します。

iSCP のメッセージングモデル (p. 14)
intdash では、エッジからサーバーに送信された時系列データを、サーバーサイドのブローカーが転送します。この転送の仕組みについて説明します。

ペイロードのパース処理 (p. 18)
時系列データのペイロードから実際のデータを取り出す処理（パース処理）は、サーバーサイドまたはクライアントサイドで行われます。このパース処理について説明します。

付録: データタイプとペイロードの定義 (p. 23)
データタイプによって異なるペイロードのフォーマットを説明します。

付録: 受信データのフィルタリング (p. 46)
intdash では、エッジが受信するデータをフィルタリングすることができます。フィルタリングの仕組みについて説明します。

02 intdash のデータモデル

ここでは、iSCP を理解するための前提として、intdash が使用するデータモデルについて解説します。

2.1 時系列データとデータポイント（時刻を付与されたペイロード）

時系列データ（time-series data）とは、時間経過に沿って取得され、時刻情報が付与された「何らかのデータ」のことです。データに時刻情報が付与されることにより、時間軸上のある一点に紐づけられるため、ある時刻の特定のデータのことを「データポイント（data point）」と表現します。

intdash は、産業用プロトコルのデータフィールドから単純な数値や文字列のデータまで、さまざまな形式のデータを扱います。これらさまざまな形式のデータを統一的に取り扱うために、intdash は「何らかのデータ」を、全て「（何らかの形式でフォーマットされた）バイト配列」とみなします。このバイト配列のことを、ペイロードと呼びます。

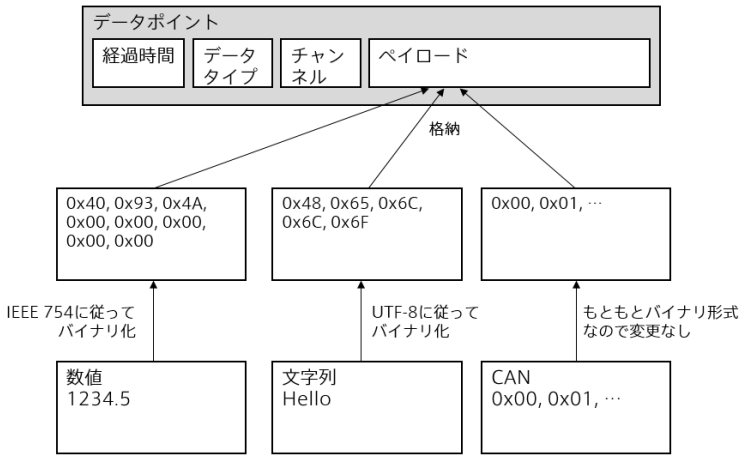


図 4　時系列データとデータポイント

注釈: ここでは、情報の正確性よりも分かりやすさを重視し、データ構造を一部簡略化して解説しています。より正確な時系列データの構造は、時系列データポイントのデータ構造 (p. 11) を参照してください。

2.2 エッジ（時系列データを送信／受信するデバイス）

intdash では、何らかのデバイス（主に、センサーを搭載した機器）で取得されたデータは、時系列データとしてリアルタイムにサーバーに送信されます。もし、そのデータの受信をリクエストしているデバイスがあれば、サーバー内のブローカーによって即座にデータが転送されます。

このような、サーバーに時系列データを送信したり、サーバーから時系列データを受信したりする存在のことを、intdash では「エッジ」と呼びます。

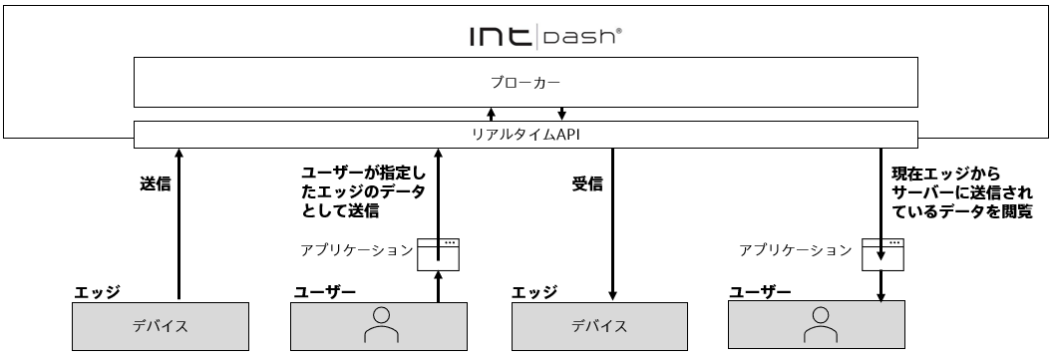


図 5　リアルタイムデータを送信／受信するエッジとユーザー

intdash では、時系列データは必ず「どれか特定のエッジのデータ」として管理されます。そのため、リアルタイム API で時系列データを送信する場合は、エッジから送信するか、またはユーザーが「特定のエッジを指定し、そのデータとして」送信する必要があります。

これに対し、リアルタイム API での受信は、intdash サーバーでの認証ができれば可能です。「エッジ」として認証しても、「ユーザー」として認証しても、リアルタイムデータを受信できます。ユーザーとして認証して受信する場合としては、Visual M2M Data Visualizer や intdash Edge Finder でリアルタイムデータを閲覧する場合が挙げられます。

2.3 計測（ある基準時刻のもとに管理される時系列データポイントの集合）

intdash は、時系列データの時刻を、基準となる時刻（基準時刻）とその基準時刻からの経過時間に分解して管理します。基準時刻と経過時間に分解する理由については、基準時刻と経過時間 (p. 10) を参照してください。

intdash では、ある基準時刻を共有した時系列データポイントの集まりを「計測（Measurement)」と呼びます。例えば、以下の図の例には、2020 年 9 月 1 日 9 時 00 分 00 秒を基準時刻として、その基準時刻からの経過時間 0.10 秒では速度 15.0km/h、経過時間 0.15 秒では 17.5km/h、経過時間 0.30 秒では 19.5km/h… といったデータポイントが存在しています。これら、1 つの基準時刻を持つ一連のデータポイントの集まりが、計測です。

図 6　計測の概念

注釈: この「計測」という名称や概念は、intdash や iSCP の規格策定段階において、アプトポッドが主たる事業としていた自動車計測が由来となっています。自動車計測などのデータ収集において、一度の計測行為で収集したデータを、計測開始時点を基準として管理したすることが多かったことから「計測」という概念が生まれました。

2.4 基準時刻と経過時間

データポイントの時刻を、絶対時刻ではなく「基準時刻」と「基準時刻からの経過時間」に分解して管理するのは、時系列データを複数の異なる時間軸上で再現可能にするためです。デバイスを介して現実世界から取得される時系列データは、絶対的な唯一の時間軸である絶対時刻に簡単に紐付けられるように思えますが、現実はもっと複雑です。デバイスが持つクロックの精度には限界があり、そのデバイスが認識している時間軸は、実は僅かに絶対時刻からずれているためです。

クロックのずれに対する一般的な対処方法は、GPS（Global Positioning System）や NTP（Network Time Protocol）などの同期手法を用いてクロックをなるべく同期させ、同期したクロックをデータへの時刻付与に使用するというものでしょう。しかし GPS や NTP などの同期手法を使っても、精度が大きく劣化する場合があります。そのため、何らかの同期手法を用いて同期したクロックであっても、それを絶対時刻としてしまうことは厳密に言えば誤りです。

一方で、「ある時点からの経過時間」であれば、同期の有無に関わらずデバイスのカウンターを使用していつでも取得することができます。経過時間さえ正しく取得できていれば、基準時刻をもとに絶対時刻を算出することで、いずれの同期手法を元にした時刻情報も、もれなく再現することが可能となります。

intdash では、このように基準時刻と経過時間に分解して時刻情報を管理するため、複数の同期手法を併用したり、あとから同期手法を切り替えて比較したり、手動で正しい時刻情報を設定し直したりすることを、容易に行うことができます。

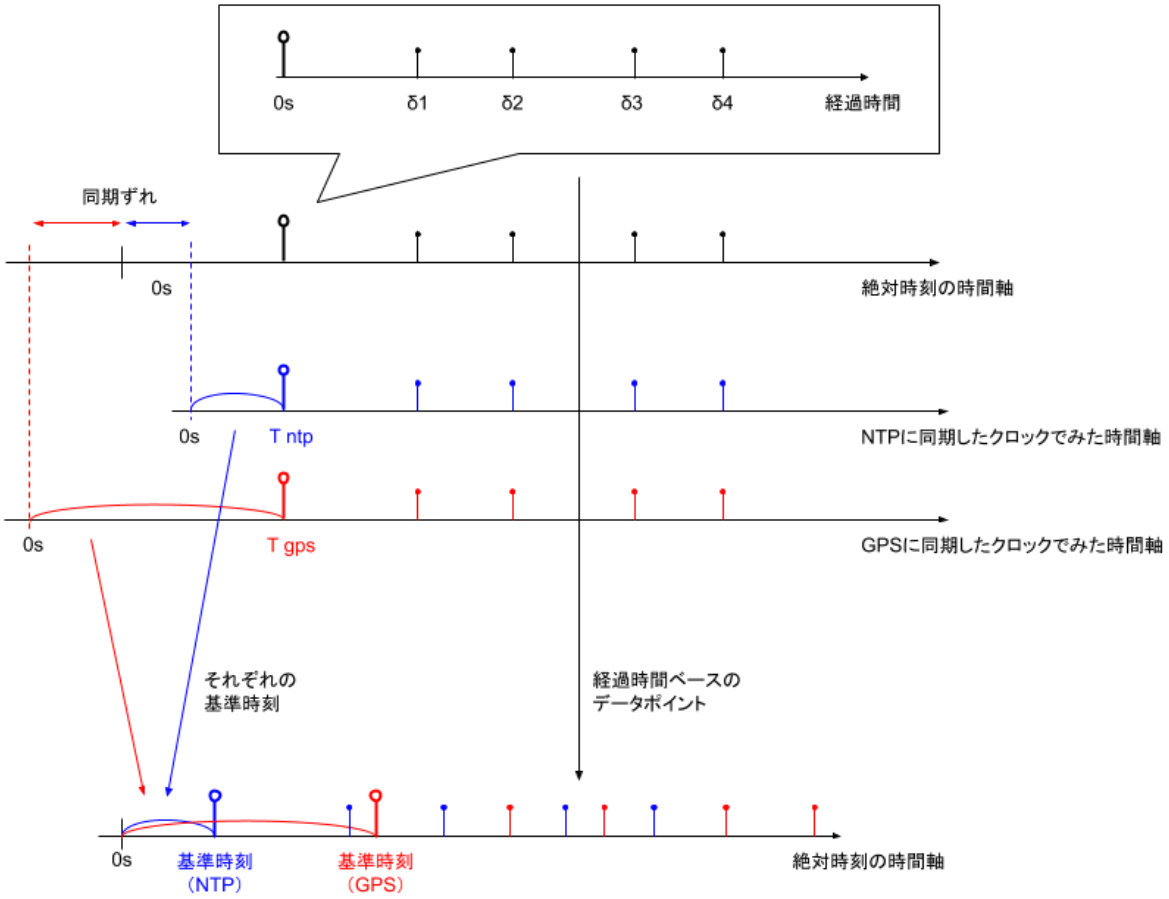


図 7　基準時刻と経過時間による絶対時刻へのマッピング

2.5 時系列データポイントのデータ構造

intdash によって管理される時系列データのデータポイントは、ヘッダーとペイロードにより構成されます。また、データタイプ、データ ID、チャンネルという識別子を持っています。

2.5.1 ヘッダー情報とペイロード

時系列データポイントは、経過時間やチャンネルなどの情報を含むヘッダーと、データ本体を表すペイロードにより構成されます。

ヘッダー情報

経過時間: データポイントの時刻を、基準時刻からの経過時間で格納します。基準時刻と経過時間 (p. 10) に記載の通り、intdash は時刻情報を基準時刻と経過時間で管理します。

チャンネル: データの識別に使用する番号です。データタイプや、データタイプごとに定義されるデータ ID とともに、複合識別子としてデータを識別します。データの識別方法は データの識別子 (p. 11) を参照してください。

データタイプ: ペイロードに格納するデータの型です。

ペイロード

データ本体のバイト配列です。ペイロードのフォーマットはデータタイプごとに異なります。データタイプごとのペイロードのフォーマットについては、付録: データタイプとペイロードの定義 (p. 23) を参照してください。

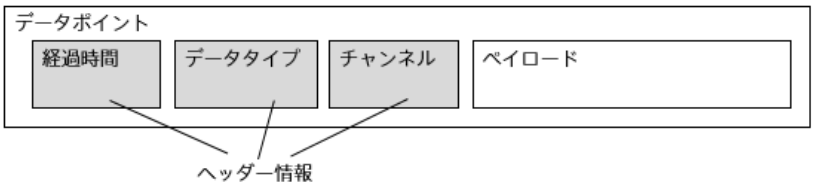


図 8　時系列データのデータ構造

2.5.2 データの識別子

時系列データは、データの型や取得元、データの種類（データが持つ意味）などにより分類されます。データの識別には、以下の 3 つの情報が複合識別子として使用されます。

- データタイプ（データの型の識別子）
- データ ID（データが持つ意味の識別子）
- チャンネル（データの取得元の識別子）

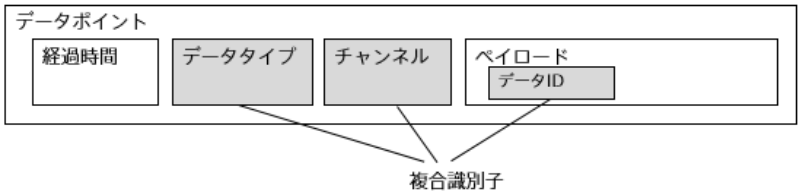


図 9　時系列データの識別子

データタイプ（データの型の識別子）

データタイプは、ペイロードに格納したデータの型（CAN、NMEA、数値、文字列など）を表します。iSCP で定義されているデータタイプとその識別番号については、付録: データタイプとペイロードの定義 (p. 23) を参照してください。

データ ID（データが持つ意味の識別子）

データ ID は、ペイロードに格納されたデータが持つ意味を表します。例えば、同じ浮動小数点型のデータであっても、それが速度を意味するのか加速度を意味するのかを区別するのがデータ ID の役割です。データ ID の割り当て方法は、データタイプごとに異なります。例えば、CAN のデータタイプの場合は、ペイロードに含まれる CAN ID がデータ ID として使用されます。JPEG や PCM などの一部のデータタイプでは、固定のデータ ID が割り当てられます。データタイプごとのデータ ID の割り当て方法は 付録: データタイプとペイロードの定義 (p. 23) を参照してください。

チャンネル（データの取得元の識別子）

同じデータタイプかつ同じデータ ID を持つデータが、複数のデバイスから取得されることがあります。このような場合のために、取得元デバイスを区別できるように与える識別子がチャンネルです。チャンネルには 0 から 255 までの数値が使用されます。

注釈: iSCP では、計測の基準時刻も 1 つのデータポイントとして送信します。intdash 内部で各データポイントの時刻を求める際には、データポイントとして送信された基準時刻に、各データポイントの経過時間を足すことにより計算します。

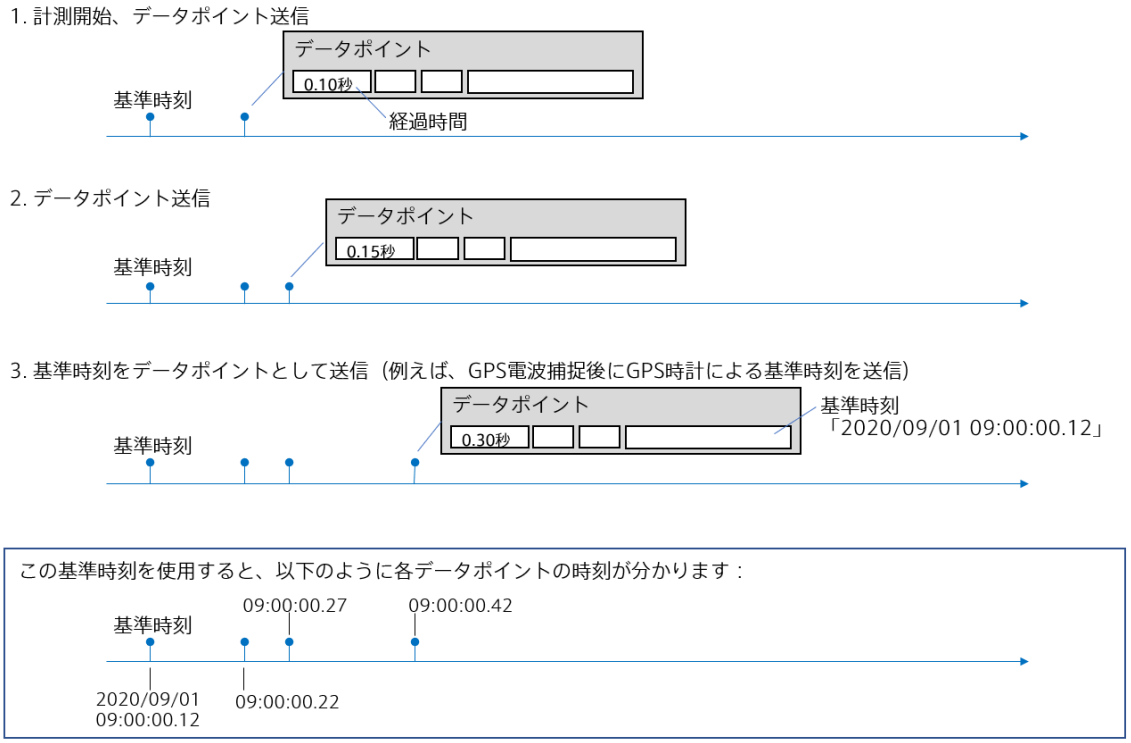


図 10　基準時刻をデータポイントとして送信する例

03 iSCP のメッセージングモデル

エッジがサーバーに時系列データを送信すると、サーバーサイドのブローカーがそのデータを転送します。このような、エッジとブローカーによるデータのやり取りは、Pub/Sub（パブリッシュ／サブスクライブ）モデルで考えることができます。

3.1 アップストリームとダウンストリーム

iSCP では、WebSocket のコネクション上に、ストリームという仮想的な通信経路を形成します。送信側のストリーム（アップストリーム）では、ストリームごとに 再送／永続化の各種フラグ (p. 17) と宛先エッジを指定します。受信側のストリーム（ダウンストリーム）では、ストリームごとにフィルタリングの条件（どの宛先を持つデータを受信するか、どのデータ識別子を持つデータを受信するか）を指定します。

ストリームはコネクション上に複数形成することができ、ストリームと各種フィルタリング条件などを組み合わせて使用することで、柔軟な送受信方法を実現することができます。

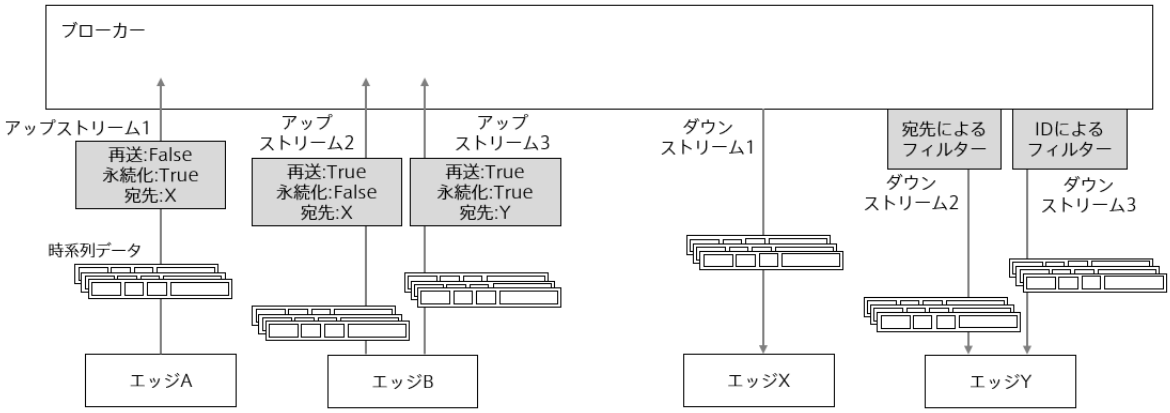


図 11　アップストリームとダウンストリームの例

3.2 データの転送

intdash のブローカーは、エッジからデータを受信すると、そのデータを他のエッジに転送します。ブローカーは、受信した 1 つのデータを複製して複数のエッジに転送することができます。これにより、エッジとエッジの間で多対多伝送を実現します。

- 送信側エッジに必要なのは、ブローカーにデータを送信することだけです。
- 受信側のエッジは、ブローカーへデータの受信要求をします。受信要求は、「エッジ A のデータが入手できたらこちらに送ってください」というように、送信側エッジを指定することで行います。
- ブローカーは、受信要求（サブスクライブ）をしているエッジにデータを転送します。

たとえば、以下の図では、送信側であるエッジ A が、ブローカーにデータを送信しています。受信側であるエッジ X は、エッジ A からのデータの受信要求をしています。もう 1 つの受信側エッジ Y もエッジ A からのデータを受信要求しています。このような場合、ブローカーは、エッジ A から送られてきたデータを複製し、受信側エッジ X と Y にそれぞれ転送します。このとき送信側エッジ A は、自分が送信したデータをどのエッジが受

信するのか、また何個のエッジが受信するのかを知る必要はありません。

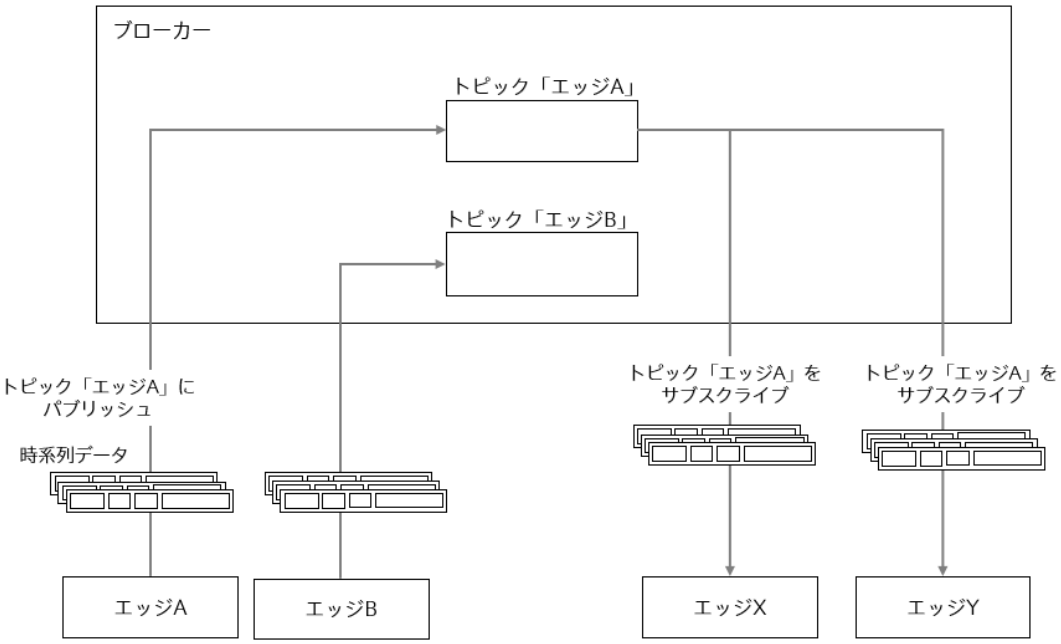


図 12　メッセージングモデル

以上が、基本のメッセージングモデルです。これは、送信側エッジの名前をトピックとした Pub/Sub メッセージングであると考えることもできます。上の例で言うと、エッジ A が「エッジ A」というトピックでデータをパブリッシュしており、エッジ X とエッジ Y はトピック「エッジ A」をサブスクライブしていると考えることができます。

3.3 宛先によるデータのフィルタリング

上述のように、受信側エッジが送信側を指定してサブスクライブするのが基本的なメッセージングモデルですが、各データの宛先を送信側から指定することもできます。そのためには、送信側のアップストリームで宛先を付与します。受信側のダウンストリームでは、自分宛のデータのみを受信するためのフィルターを設定することができます。

例えば、エッジ A が、エッジ X またはエッジ Y に宛てたデータを送信し、エッジ X とエッジ Y は自分宛のデータのみを受信したい場合、次のような方法をとります。

- エッジ A からサーバーへのアップストリーム
  - エッジ X に送るデータのアップストリームには、宛先としてエッジ X を指定します。
  - エッジ Y に送るデータのアップストリームには、宛先としてエッジ Y を指定します。
- サーバーからエッジ X へのダウンストリーム
  - エッジ A からのデータを受信要求します。また、宛先がエッジ X（自分）となっているデータのみを受信するフィルターを設定します。
- サーバーからエッジ Y へのダウンストリーム
  - エッジ A からのデータを受信要求します。また、宛先がエッジ Y（自分）となっているデータのみを受信するフィルターを設定します。

こうすることで、エッジ X とエッジ Y は自分宛のデータのみを受信することができます。

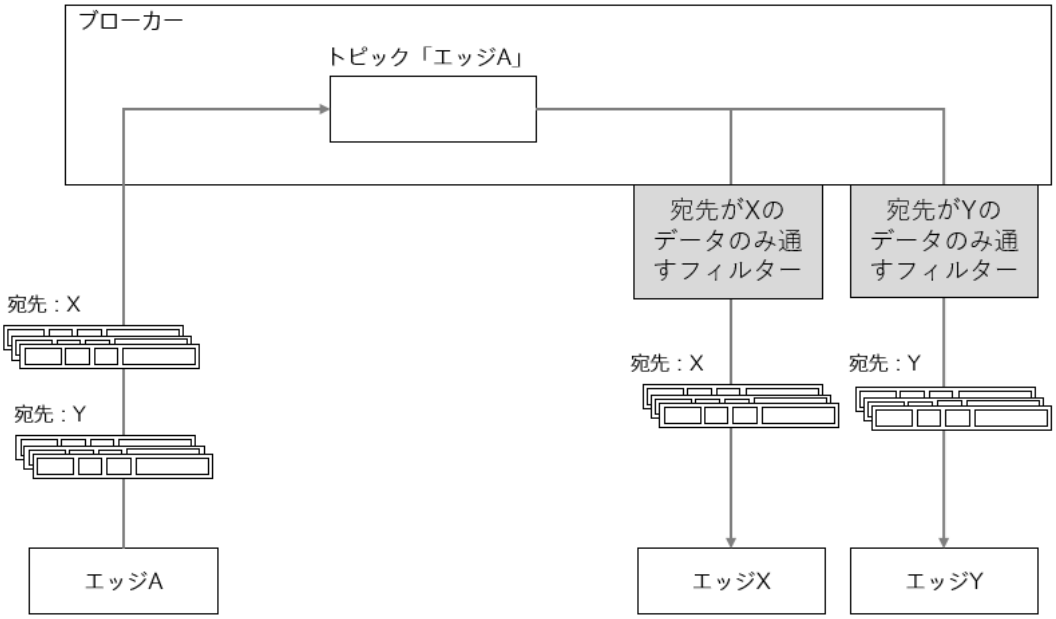


図 13　宛先によるフィルタリング

3.4 データ識別子によるフィルタリング（ダウンストリーム）

受信時のフィルタリングには、データの識別子 (p. 11) も使用することができます。データ識別子によるフィルタリングを行うと、「特定のチャンネル、かつ特定のデータタイプ、かつ特定のデータ ID」のデータのみ受信することができます。

データタイプごとにデータ ID の形式は異なります。各データタイプのデータ ID については、付録: データタイプとペイロードの定義 (p. 23) を参照してください。

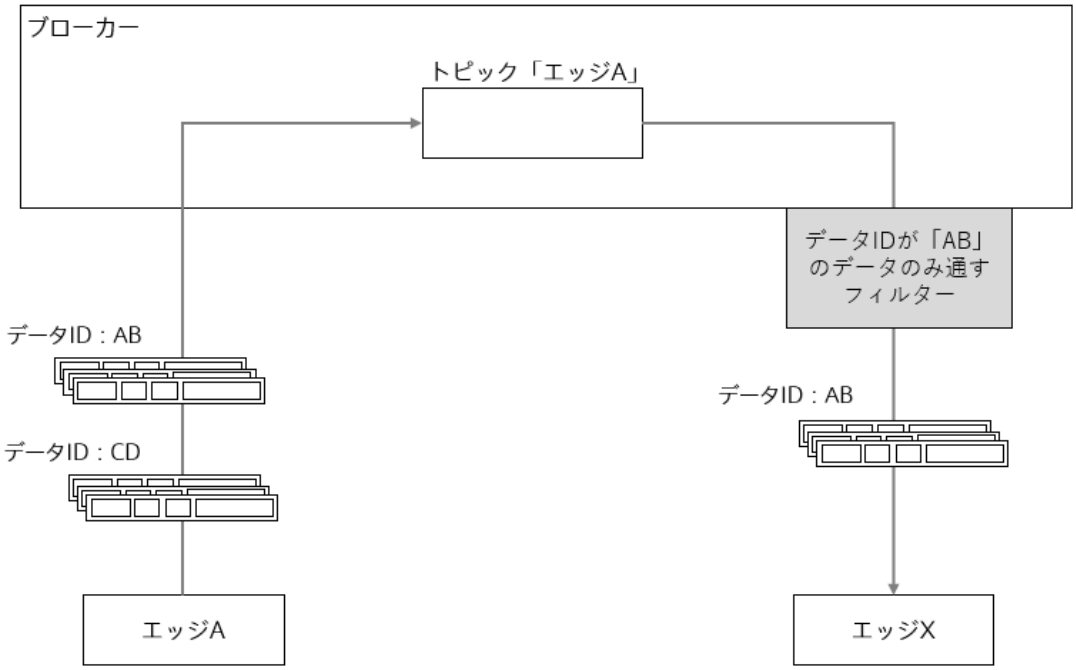


図 14　データ識別子によるフィルタリング

3.5 再送フラグと永続化フラグ（アップストリーム）

データを送信する際、送信側のエッジでは、ブローカーによるデータの扱いを 2 つのフラグにより制御することができます。

再送フラグ

データが欠損した場合、通常は自動的に再送処理が行われるため、同じデータが複数回送信される場合があります。送信側エッジは、欠損データの再送を行う場合は再送フラグをオンにします。ブローカーは、再送フラグがオンのデータは他のエッジに転送しません。

永続化フラグ

通常は、伝送されたデータはすべて永続化されますが、ユースケースによってはデータを永続化したくない場合もあります。そのような場合には、送信側エッジでアップストリームの永続化フラグをオフにしてデータを送信します。ブローカーは、永続化フラグがオフのデータは永続化しません。

04 ペイロードのパース処理

General Sensor（アプトポッド独自定義）(p. 36) など一部のデータタイプでは、ペイロードの中に、複数の値データが格納されることがあります。このようなデータをアプリケーションで使用するためには、ペイロードから値データを抽出するパース処理が必要です。

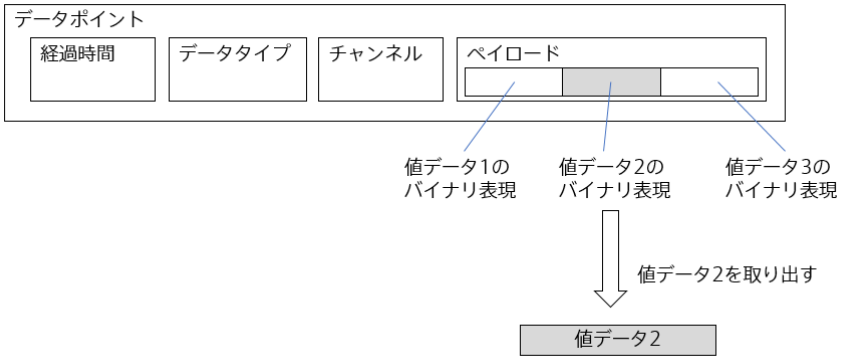


図 15　パース処理

値データを抽出するパース処理は、サーバーサイドで行うものと、クライアントサイドで行うものとがあります。

4.1 サーバーサイドにおけるパース処理

intdash SDK for Python を使う場合など、クライアントサイドでの処理パフォーマンスが十分でない場合には、サーバーサイドでパース処理を行うことができます。サーバーサイドでパース処理を実施するには、パースの方法を定めた「信号定義（Signal）」という情報をサーバーに登録しておく必要があります。

信号定義に従ってサーバーサイドでパースが行われ、データが取り出されると、そのデータは新たな時系列データポイントとなります。

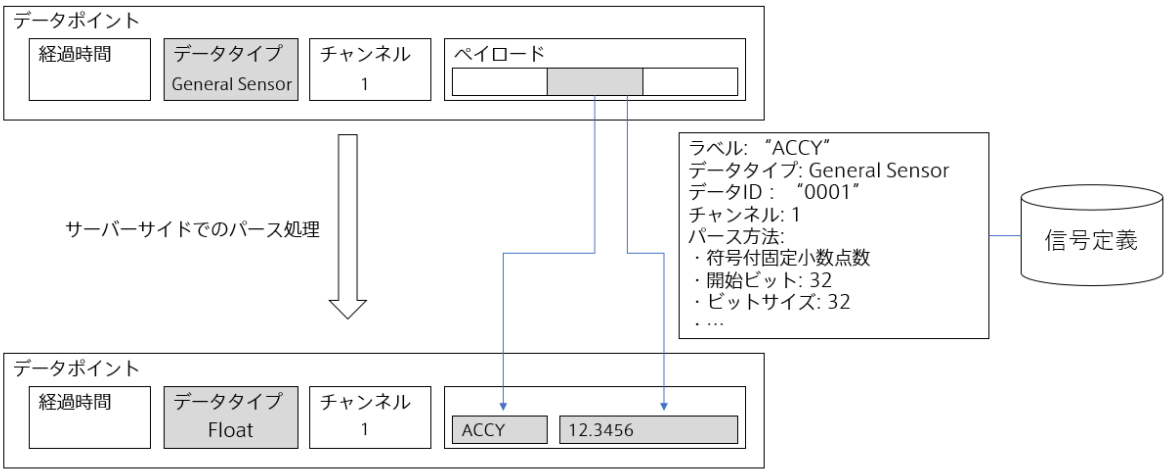


図 16　サーバーサイドにおけるパース処理の例

4.1.1 信号定義

信号定義を構成する主な情報は以下の通りです。

- データ識別子（チャンネル、データタイプ、データ ID）
- ラベル
- パース方法

データ識別子（チャンネル、データタイプ、データ ID）は、パースの対象となるデータを特定するために使用されます。

データ識別子で特定されたデータは、指定されたパース方法に則ってパースされます。パースの結果として得られたデータは、新たな時系列データポイントのペイロードに格納されます。データポイントのデータタイプは、Bytes、Float、または String となります。信号定義が持つラベルは、パース後のデータポイントのデータ ID として使用されます。

4.1.2 パース方法

パース方法には、以下の種類があります。

変換の方法	説明	変換後のデータタイプ
変換なし（ `none` ）	変換をしないことを表します。元のペイロードを Bytes データとしてそのまま格納します。	Bytes
浮動小数点数（ `num_floating_point` ）	ペイロードの一部を切り出し、32bit または 64bit の浮動小数点数として読み出します。	Float
固定小数点数（ `num_fixed_point` ）	ペイロードの一部を切り出し、固定小数点数として読み出します。読み出し方法は、CAN データフィールドの読み出し方法に準じます。詳細は （参考）CAN データフィールドの読み出し方法 (p. 20) を参照してください。	Float
バイト列切り出し（ `sub_bytes` ）	ペイロードの一部を切り出します。	Bytes
文字列切り出し（ `sub_string` ）	ペイロードの一部を切り出ます。	String
CSV（ `csv` ）	ペイロードを CSV 形式の文字列とみなしてその中から 1 フィールドを取り出します。	String または Float
JSON（ `json` ）	ペイロードを JSON 形式の文字列とみなしてその中から 1 フィールドを取り出します。	String または Float

4.1.3 （参考）CAN データフィールドの読み出し方法

CAN のデータフィールドは最大 8 バイトのバイト配列です。これを読み出すには、以下の手順で変換処理を行います。

1. 開始ビットから終了ビットまでのビット列を切り出します。

   CAN データフィールドのビットの並びには、リトルエンディアンとビッグエンディアンの 2 種類が存在します。CAN データからデータを切り出す際には以下のように考えて、信号定義の `start bit` と `length` を指定してください。

   まずバイト列の先頭ビットから末尾ビットまでに連番を付与します。この連番をもとに、`start bit` と `length` を指定してビットを切り出します。例えば、`start bit` が 3、`length` が `10` の場合、リトルエンディアンとビッグエンディアンではそれぞれ以下の枠のついたビットが切り出されます。

	MSB							LSB
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	0	1	2	3	4	5	6	7
byte 1	8	9	10	11	12	13	14	15
…	…	…	…	…	…	…	…	…
byte 7	56	57	58	59	60	61	62	63

図 17　リトルエンディアンのデータの例

	LSB							MSB
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
byte 0	0	1	2	3	4	5	6	7
byte 1	8	9	10	11	12	13	14	15
…	…	…	…	…	…	…	…	…
byte 7	56	57	58	59	60	61	62	63

図 18　ビッグエンディアンのデータの例

2. 切り出したビット列を符号付き整数または符号なし整数として整数に変換します。
3. 以下の数式に従って、定数倍と定数オフセットを適用します。

   変換後の値 = scale × (手順 2 で得た整数値) + offset

   - `scale`: 定数倍用の定数
   - `offset`: 定数オフセット用の定数

4.2 Visual M2M Data Visualizer におけるパース処理

ウェブアプリケーションである Visual M2M Data Visualizer（以下、Data Visualizer）は、クライアントサイドでパース処理を行います。Data Visualizer でパースを行うには、.dat 形式のパース定義ファイルが必要です。

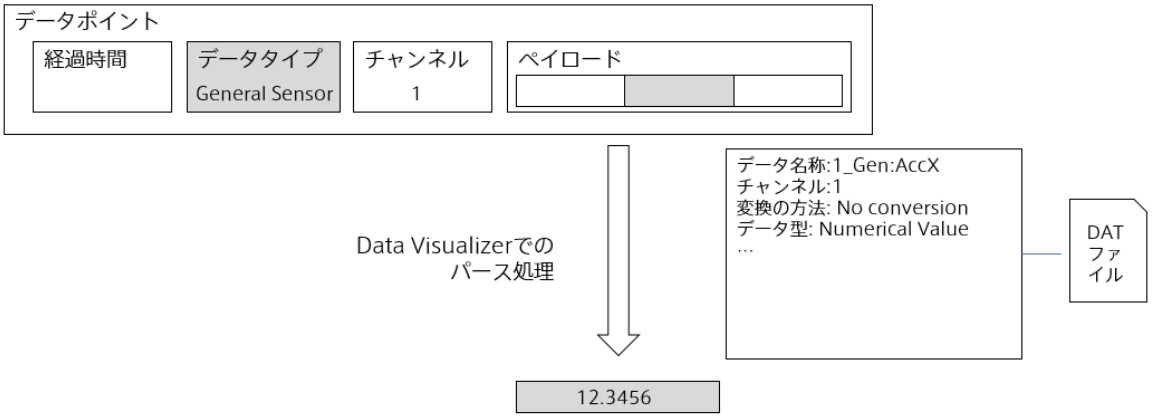


図 19　Data Visualizer におけるパース処理の例

4.2.1 パース方法の種類

Data Visualizer におけるパースの方法には、以下の種類があります。

変換の方法	データ型指定	Data Name へのサフィックス指定	説明
No conversion	Character String	-	サーバーサイドにおけるパース処理の「文字列切り出し」に相当
No conversion	Numerical Value または Character String	:f32	サーバーサイドにおけるパース処理の「浮動小数点数」に相当
No conversion	Numerical Value または Character String	:f64	サーバーサイドにおけるパース処理の「浮動小数点数」に相当
Binary Map	Numerical Value または Character String	-	サーバーサイドにおけるパース処理の「固定小数点数」に相当
CSV	Numerical Value または Character String	-	サーバーサイドにおけるパース処理の「CSV」に相当
JSON	Numerical Value または Character String	-	サーバーサイドにおけるパース処理の「JSON」に相当

05 付録: データタイプとペイロードの定義

iSCP では、以下のデータタイプが定義されています。

データタイプを示すコード	データタイプ名	説明	データ ID の文字列表現
0x01 (1)	CAN (p. 32)	CAN	%08x
0x02 (2)	NMEA (p. 34)	NMEA 文字列	%s
0x03 (3)	General Sensor (p. 36)	汎用センサー	%04x
0x04 (4)	Controlpad (p. 42)	汎用コントロールパッド	%02x
0x05 (5)	MAVLink 1 Packet (p. 33)	MAVLink 1 パケット（Micro Air Vehicle/ドローン用の通信プロトコル）	%02x-%02x-%02x-%02x
RESERVED	...	...	
0x09 (9)	JPEG (p. 28)	JPEG	jpeg
0x0A (10)	String (p. 25)	文字列（プリミティブデータ）	%s
0x0B (11)	Float (p. 26)	倍精度浮動小数点数（プリミティブデータ）	%s
0x0C (12)	Int (p. 27)	64bit 符号付き整数（プリミティブデータ）	%s
0x0D (13)	H.264 (p. 28)	H.264	%02x
0x0E (14)	Bytes (p. 23)	バイト列（プリミティブデータ）	%s
0x0F (15)	PCM (p. 30)	PCM（WAVE）	pcm
0x010 (16)	AAC (p. 31)	AAC（ADTS）	aac
RESERVED	...	...	-
0x7F (127)	Generic (p. 43)	汎用バイナリデータ	%04x

5.1 Bytes（文字列 ID を持ったバイト列データ）

5.1.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| IDLength      | ID                                            |
+---------------+---------------+---------------+-----------//--+
| Data                                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
IDLength	1	-	なし	ID フィールドの長さ
ID	IDLength	-	-	UTF-8 エンコードされたデータ ID
Data	-	-	-	データを格納したバイト列

5.1.2 データ ID

- ID フィールドの内容をデータ ID とする
- データ ID の文字列表現: ID フィールドのバイト列を UTF-8 でデコードした結果の文字列（%s）

5.1.3 JSON での表現

項目	種別	必須	内容
i	string	yes	ID
d	string	yes	Data の Base64 表現

例:

```
{
  "i": "128",
  "d": "8PHy8/T19vc="
}
```

5.2 String（文字列 ID を持った文字列データ）

5.2.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID Length     | ID                                            |
+---------------+---------------+---------------+-----------//--+
| Data                                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
IDLength	1	-	なし	ID フィールドの長さ
ID	IDLength	-	-	UTF-8 エンコードされたデータ ID
Data	-	-	-	文字列データを UTF-8 エンコードした結果のバイト列

5.2.2 データ ID

- ID フィールドの内容をデータ ID とする
- データ ID の文字列表現: ID フィールドのバイト列を UTF-8 でデコードした結果の文字列（%s）

5.2.3 JSON での表現

項目	種別	必須	内容
i	string	yes	ID
d	string	yes	Data フィールドを UTF-8 でデコードした結果の文字列

例:

```
{
  "i": "128",
  "d": "abcdefghijk"
}
```

5.3 Float（文字列 ID を持った倍精度浮動小数点数）

フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID Length     | ID                                            |
+---------------+---------------+---------------+-----------//--+
| Data                                                          |
+---------------+---------------+---------------+---------------+
|                                                               |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
IDLength	1	-	なし	ID フィールドの長さ
ID	IDLength	-	-	UTF-8 エンコードされたデータ ID
Data	8	LE	-	倍精度浮動小数点数を格納したバイト列[1]

5.3.1 データ ID

- ID フィールドの内容をデータ ID とする
- データ ID の文字列表現: ID フィールドのバイト列を UTF-8 でデコードした結果の文字列（%s）

5.3.2 JSON での表現

項目	種別	必須	内容
i	string	yes	ID
d	number	yes	データ値

例:

```
{
  "i": "128",
  "d": 12345.6789
}
```

[1] 倍精度浮動小数点数を格納したバイト列。IEEE 754 に準拠したフォーマットを使用する。

5.4 Int（文字列 ID を持った 64bit 符号付き整数）

5.4.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID Length     | ID                                            |
+---------------+---------------+---------------+-----------//--+
| Data                                                          |
+---------------+---------------+---------------+---------------+
|                                                               |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
IDLength	1	-	なし	ID フィールドの長さ
ID	IDLength	-	-	UTF-8 エンコードされたデータ ID
Data	8	LE	あり	64bit 符号付き整数を格納したバイト列

5.4.2 データ ID

- ID フィールドの内容をデータ ID とする
- 文字列表現: ID フィールドのバイト列を UTF-8 でデコードした結果の文字列（%s）

5.4.3 JSON での表現

項目	種別	必須	内容
i	string	yes	ID
d	number	yes	データの値

例:

```
{
  "i": "128",
  "d": 123456789
}
```

5.5 JPEG（JPEG 画像）

フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| Data                                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
Data	-	-	-	JPEG (ISO/IEC10918-1, Annex B) のバイナリデータ

5.5.1 データ ID

- "jpeg"で固定とする
- 文字列表現: "jpeg"で固定とする

5.5.2 JSON での表現

項目	種別	必須	内容
d	string	yes	Data の Base64 表現

例:

```
{
  "d": "/9j/2wCEAAQCAwMDAgQDAwME"
}
```

5.6 H.264（H.264/AVC 動画）

5.6.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| Type          | Data                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
Type	1	-	なし	Data 部の種別を表す番号
Data	-	-	-	Type に応じたデータ

Data フィールドには、Type に応じて以下のデータを格納します。

Type	区分	Data に格納するデータ
0x00	H.264 データ	IDR slice (nal_unit_type == 5) である NALunits の 1 フレーム分[2] の連結[3]
0x01	H.264 データ	non-IDR slice (nal_unit_type == 1) である NALunits の 1 フレーム分の連結
0x08	H.264 データ	H.264 のデコードに必要な NAL units[4] の連結。Type 0x00 の都度、先行して 1 つ生成するものとする
0x09	H.264 データ	Supplemental Enhancement Information(nal_unit_type == 6) である NAL unit
0x81	メタ情報	回転情報[5] を格納する。Type 0x00 の都度、先行して 1 つ生成するものとする

5.6.2 データ ID

- Type フィールドをデータ ID とする
- 文字列表現: Type を 0 埋め小文字 2 桁の 16 進数として文字列化する (%02x)

5.6.3 JSON での表現

項目	種別	必須	内容
t	number	yes	Type
d	string	yes	Data の Base64 表現

例:

```
{
  "t": 0,
  "d": "AAAAAWdCQCi7QCgC2/gK..."
}
```

[2] 1 フレーム分とは、nal_unit_type が 1-5 (Coded slice) の時、同一の nal_unit_type の NAL unit について、slice_header の first_mb_in_slice が 0 であるものから、次に 0 であるものの前までの NAL units を指します。

[3] 「ITU-T Rec. H.264 ¦ ISO/IEC 14496-10 Advanced Video Coding」における「Byte stream format (Annex B)」に従い、以下のように連結すること。`[start code prefix] + [NAL unit] + [start code prefix] + [NAL unit] + [start code prefix] + [NAL unit]`

[4] H.264 のデコードに必要な NAL units とは、SPS (nal_unit_type == 7) と PPS (nal_unit_type == 8) の NAL units を指します。

[5] 1 バイトの符号なし整数で、画面の回転方向を格納します (0:上、1:下、2:左、3:右)。

5.7 PCM（WAVE 形式の音声波形情報）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| FormatID                      | Channels                      |
+---------------+---------------+---------------+---------------+
| SampleRate                                                    |
+---------------+---------------+---------------+---------------+
| BitPerSample  | Data                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
FormatID	2	LE	なし	WAVE[-6] で定義されたフォーマット ID
Channels	2	LE	なし	格納される音声のチャンネル数
SampleRate	4	LE	なし	サンプリング周波数 [Hz]
BitPerSample	1	-	なし	ビットレート [bit/sample]
Data	-	-	-	WAVE[-6] に準拠する波形情報

5.7.1 データ ID

- "pcm"で固定とする
- 文字列表現: "pcm"で固定とする

5.7.2 JSON での表現

項目	種別	必須	内容
f	number	yes	FormatID
c	number	yes	Channels
r	number	yes	SampleRate
b	number	yes	BitPerSample
d	string	yes	Data の Base64 表現

例:

```
{
  "f": 1,
  "c": 2,
  "r": 44100,
  "b": 16,
```

(次のページに続く)

[-6] 音声データ用コンテナーフォーマットの WAVE (RIFF waveform Audio Format)

(前のページからの続き)

```
  "d": "AAAAAWdCQCi7QCgC2/gK..."
}
```

5.8 AAC（ADTS 形式の AAC 音声）

5.8.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| Data                                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
Data	-	-	-	ADTS フレーム[7]

5.8.2 データ ID

- "aac"で固定とする
- 文字列表現: "aac"で固定とする

5.8.3 JSON での表現

項目	種別	必須	内容
d	string	yes	Data の Base64 表現

例:

```
{
  "d": "/9j/2wCEAAQCAwMDAgQDAwME"
}
```

[7] 1 つの ADTS ヘッダーおよび CRC と Raw Data Blocks（「MPEG-2 AAC (ISO/IEC 13818-7)」もしくは「MPEG-4 AAC (ISO/IEC 14496-3, Subpart 4)」に従うバイナリデータ）からなる ADTS フレーム。CRC は、ADTS ヘッダーのフィールド値によっては、省略される場合がある。

5.9 CAN（Controller Area Network 用の通信プロトコル）

5.9.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID                                                            |
+---------------+---------------+---------------+---------------+
| DLC           | Data                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
ID	4	LE	なし	CAN ID
DLC	1	-	なし	DLC
Data	DLC	-	-	CAN のデータフィールド

5.9.2 データ ID

- ID フィールドに格納された CAN ID をデータ ID とする
- 文字列表現: ID フィールドの値を 0 埋め小文字 8 桁の 16 進数として文字列化する (%08x)

5.9.3 JSON での表現

項目	種別	必須	内容
i	number	yes	ID
d	string	yes	Data の Base64 表現

例:

```
{
  "i": 128,
  "d": "8PHy8/T19vc="
}
```

5.10 MAVLink 1 Packet（Micro Air Vehicle/ドローン用の通信プロトコル）

5.10.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| STX           | LEN           | SEQ           | SYS ID        |
+---------------+---------------+---------------+---------------+
| COMP ID       | MSG ID        | PAYLOAD                       |
+---------------+---------------+---------------+---------------+
|                               | CHECKSUM                      |
+---------------+-----------//--+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
STX (PacketStart Marker)	1	-	なし	パケット開始を表すフラグ (0xFD)
LEN (PayloadLength)	1	-	なし	ペイロード長
SEQ (PacketSequence)	1	-	なし	コンポーネントごとにインクリメントされるカウンタ
SYS ID (SystemID)	1	-	なし	System ID
COMP ID(Component ID)	1	-	なし	Component ID
MSG ID(Message ID)	1	-	なし	Message ID
PAYLOAD	LEN	-	-	ペイロード
CHECKSUM	2	LE	-	STX を除いた部分のチェックサム (CRC-16-CCITT)

各フィールドに格納する値は、MAVLink 1 Packet Format に従う

5.10.2 データ ID

- 下記 4 フィールドを合わせてデータ ID とする
  - STX、SYS ID、COMP ID、MSG ID
- 文字列表現: 上記 4 フィールドを 0 埋め小文字 2 桁の 16 進数として文字列化し、ハイフンで結合する (%02x-%02x-%02x-%02x)

5.10.3 JSON での表現

項目	種別	必須	内容
stx	number	yes	PacketStartSign
seq	number	yes	PacketSequence
sys	number	yes	SystemID
comp	number	yes	ComponentID
msg	number	yes	MessageID
chk	string	yes	CheckSum の Base64 表現
data	string	yes	Data の Base64 表現

例:

```
{
  "stx": 254,
  "seq": 123,
  "sys": 1,
  "comp": 200,
  "msg": 2,
  "chk": "AQE=",
  "data": "ADUIKclXBQCghgEA"
}
```

5.11 NMEA（GPS データの通信用プロトコル）

5.11.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| NMEAString                                                    |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
NMEAString	-	-	-	NMEA 文字列

5.11.2 データ ID

- NmeaString 内のトーカ 2 文字とメッセージ 3 文字の合計 5 文字を UTF-8 エンコードしたもの
- 文字列表現 : 上記を文字列解釈する (%s)

5.11.3 JSON での表現

項目	種別	必須	内容
i	string	yes	ID を UTF-8 でデコードした結果の文字列
s	string	yes	NmeaString

例:

```
{
  "i": "GPRMC",
  "s": "$GPRMC,085120.307,A,3541.1493,N,13945.3994,E,000.0,240.3,181211,,,A*6A"
}
```

5.12 General Sensor（アプトポッド独自定義）

5.12.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID                            | Data                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
ID	2	LE	なし	数値 ID
Data	-	-	-	所定のフォーマットに従ったセンサーデータ。Data フィールドのフォーマット (p. 37) を参照してください。

5.12.2 データ ID

- ID フィールドに格納された 数値 ID をデータ ID とする
- 文字列表現: ID フィールドの値を 0 埋め小文字 4 桁の 16 進数として文字列化する (%04x)

5.12.3 JSON での表現

項目	種別	必須	内容
i	number	yes	ID
d	string	yes	Data の Base64 表現

例:

```
{
  "i": 128,
  "d": "8PHy8/T19vc="
}
```

5.12.4 Data フィールドのフォーマット

General Sensor タイプのデータは、データ ID ごとに Data フィールドのフォーマットが異なります。

ID	名称
0x0001	加速度
0x0002	加速度（重力加速度含む）
0x0003	重力加速度
0x0004	回転速度
0x0005	回転角度
0x0006	位置情報（座標）
0x0007	位置情報（高度）
0x0008	位置情報（精度）
0x0009	位置情報（方角）
0x000A	位置情報（移動速度）

加速度

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ax                                                            |
+---------------+---------------+---------------+---------------+
| ay                                                            |
+---------------+---------------+---------------+---------------+
| az                                                            |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
ax	4	LE	あり	X 軸方向加速度
ay	4	LE	あり	Y 軸方向加速度
az	4	LE	あり	Z 軸方向加速度

ax、ay、az は、生の数値 [m/s2] を 1e6 倍したものを符号付き整数で格納します。

加速度（重力加速度含む）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| agx                                                           |
+---------------+---------------+---------------+---------------+
| agy                                                           |
+---------------+---------------+---------------+---------------+
| agz                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
agx	4	LE	あり	X 軸方向加速度（重力加速度含む）
agy	4	LE	あり	Y 軸方向加速度（重力加速度含む）
agz	4	LE	あり	Z 軸方向加速度（重力加速度含む）

agx、agy、agz は、生の数値 [m/s2] を 1e6 倍したものを符号付き整数で格納します。

重力加速度

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| gx                                                            |
+---------------+---------------+---------------+---------------+
| gy                                                            |
+---------------+---------------+---------------+---------------+
| gz                                                            |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
gx	4	LE	あり	X 軸方向重力加速度
gy	4	LE	あり	Y 軸方向重力加速度
gz	4	LE	あり	Z 軸方向重力加速度

gx、gy、gz は、生の数値 [m/s2] を 1e6 倍したものを符号付き整数で格納します。

回転速度

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| rra                                                           |
+---------------+---------------+---------------+---------------+
| rrb                                                           |
+---------------+---------------+---------------+---------------+
| rrg                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
rra	4	LE	あり	正面上方向を中心軸とした回転速度（ヨーレート）
rrb	4	LE	あり	正面右方向を中心軸とした回転速度（ピッチレート）
rrg	4	LE	あり	正面方向を中心軸とした回転速度（ロールレート）

rra、rrb、rrg は、生の数値 [deg/s] を 1e5 倍したものを格納します。

回転角度

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| oaa                                                           |
+---------------+---------------+---------------+---------------+
| oab                                                           |
+---------------+---------------+---------------+---------------+
| oag                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
oaa	4	LE	あり	正面上方向を中心軸とした回転角度（ヨー角）
oab	4	LE	あり	正面右方向を中心軸とした回転角度（ピッチ角）
oag	4	LE	あり	正面方向を中心軸とした回転角度（ロール角）

oaa、oab、oag は、生の数値 [deg] を 1e6 倍したものを格納します。

位置情報（座標）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| lat                                                           |
+---------------+---------------+---------------+---------------+
| lng                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
lat	4	LE	あり	緯度
lng	4	LE	あり	経度

lat、lng は、生の数値 [deg] を 1e7 倍したものを格納します。

位置情報（高度）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| alt                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
alt	4	LE	あり	高度

alt は、生の数値 [m] を 1e3 倍したものを格納します。

位置情報（精度）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| aoc                                                           |
+---------------+---------------+---------------+---------------+
| aoa                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
aoc	4	LE	あり	座標の精度
aoa	4	LE	あり	高度の精度

aoc、aoa は、生の数値 [m] を 1e3 倍したものを格納します。

位置情報（方角）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| head                                                          |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
head	4	LE	あり	方角

head は、生の数値 [deg] を 1e6 倍したものを格納します。

位置情報（移動速度）

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| spd                                                           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
spd	4	LE	あり	移動速度

spd は、生の数値 [km/h] を 1e3 倍したものを格納します。

5.13 Controlpad（アプトポッド独自定義）

5.13.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID            | Data                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
ID	1	LE	なし	数値 ID
Data	-	-	-	所定のフォーマットに従ったコントローラ情報。各種コントローラーデータのフォーマット (p. 43) を参照してください。

5.13.2 データ ID

- ID フィールドに格納された数値 ID をデータ ID とする
- 文字列表現: ID フィールドの値を 0 埋め小文字 2 桁の 16 進数として文字列化する (%02x)

5.13.3 JSON での表現

項目	種別	必須	内容
i	number	yes	ID
d	string	yes	Data の Base64 表現

例:

```
{
  "i": 1,
  "d": "AQEBAQEBAQEBAQEBAQEBAQEBAQE="
}
```

5.13.4 各種コントローラーデータのフォーマット

ID ごとの Data フィールドのフォーマットは以下の通りです。

軸とボタン

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| axis0                         | axis1                         |
+---------------+---------------+---------------+---------------+
| axis2                         | axis3                         |
+---------------+---------------+---------------+---------------+
| axis4                         | axis5                         |
+---------------+---------------+---------------+---------------+
| axis6                         | axis7                         |
+---------------+---------------+---------------+---------------+
| bg0           | bg1           | bg2           | bg3           |
+---------------+---------------+---------------+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
axis0	2	LE	あり	軸 0
…	-	-	-	-
axis7	2	LE	あり	軸 7
bg0	1	-	-	ボタングループ 0
…	-	-	-	-
bg3	1	-	-	ボタングループ 3

- axisN は、生の数値を 1e6 倍したものを格納します。
- ボタン 0〜7 を「ボタングループ 0」（ボタン 7 が最上位ビット）、ボタン 8〜15 を「ボタングループ 1」（ボタン 15 が最上位ビット）のように、8 つのボタンごとのボタングループとします。
- bgN のボタンの状態の値は、1: ON 状態、0: OFF 状態とします。

5.14 Generic（アプトポッド独自定義）

5.14.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| ID                                                            |
+---------------+---------------+---------------+---------------+
| Data                                                          |
+---------------+---------------+---------------+-----------//--+
```

フィールド名	バイト長	エンディアン	符号	説明
ID	4	LE	なし	数値 ID
Data	-	-	-	任意のデータ

5.14.2 データ ID

- ID フィールドに格納された 数値 ID をデータ ID とする
- 文字列表現: ID フィールドの値を 0 埋め小文字 8 桁の 16 進数として文字列化する (%08x)

5.14.3 JSON での表現

項目	種別	必須	内容
i	number	yes	ID
d	string	yes	Data の Base64 表現

例:

```
{
  "i": 128,
  "d": "8PHy8/T19vc="
}
```

5.15 Base Time - 基準時刻

5.15.1 フォーマット

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
| Type          | BaseTimeSec                                   |
+---------------+---------------+---------------+---------------+
|               | BaseTimeNano                                  |
+---------------+---------------+---------------+---------------+
|               |
+---------------+
```

フィールド名	バイト長	エンディアン	符号	説明
Type	1	-	なし	基準時刻の取得元を表す番号。基準時刻の取得元 (p. 45) を参照してください。
BaseTimeSec	4	LE	なし	基準時刻の UNIX エポック経過秒 (秒部分)
BaseTimeNano	4	LE	なし	基準時刻の UNIX エポック経過秒 (ナノ秒部分)

5.15.2 基準時刻の取得元

基準時刻の取得元に応じて、以下のいずれかを使用します。

Type	名称	説明
0x01	EdgeRTC	Edge の時計により取得したもの
0x02	NTP	NTP により取得したもの
0x03	GPS	GPS 時刻より取得したもの
0xFD	Volatile	計算処理等により一時的に生成されたもの
0xFE	APIFirstReceived	計測に所属する最初の時系列データの受信時刻を基準としたもの
0xFF	Manual	手動で取得したもの

5.15.3 JSON での表現

項目	種別	必須	内容
t	number	yes	Type
b	string	yes	基準時刻の RFC3339 表記

例:

```
{
  "t": 255,
  "b": "2016-07-11T19:32:45.123456789Z09:00"
}
```

06 付録: 受信データのフィルタリング

データ識別子によるフィルタリング（ダウンストリーム）(p. 16) に記載の通り、ダウンストリームではデータ識別子（チャンネル、データタイプ、データ ID）を使って受信データをフィルタリングすることができます。受信側エッジは、フィルターを通過したデータだけを受信します。

フィルターは、各ストリームで、チャンネルとデータタイプの組み合わせに対して設定します。1 つのフィルターには複数のフィルター条件を設定することができます。

また、一致する「チャンネルとデータタイプの組み合わせ」のフィルターが存在しない場合は、データはブロックされます。

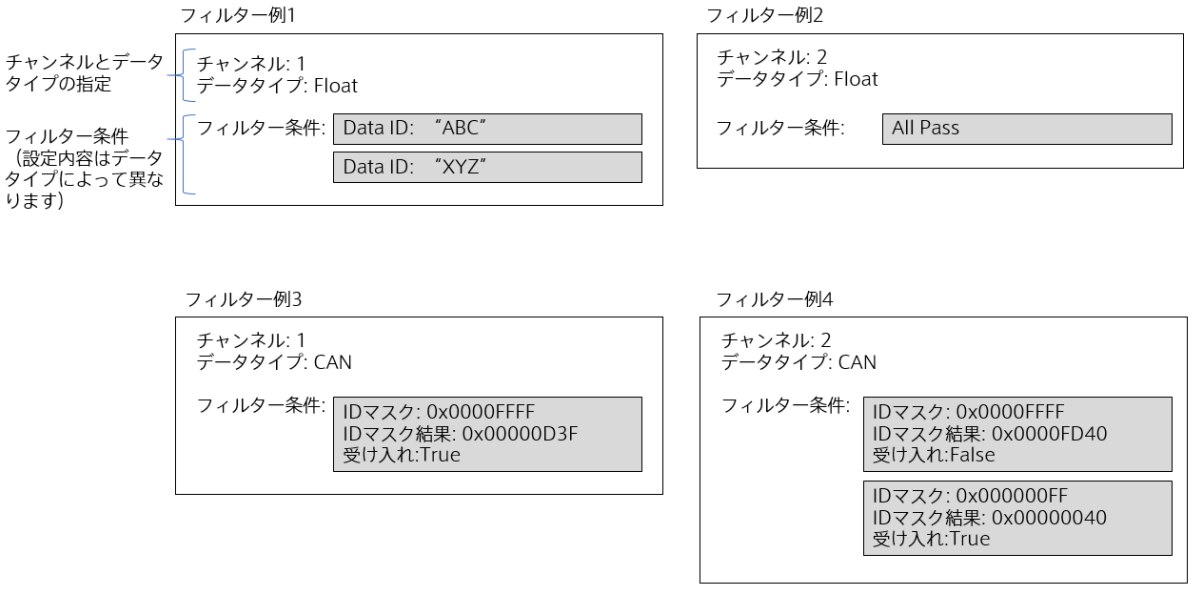


図 20　フィルターの構造

フィルター条件の設定方法は、データタイプによって異なります。詳細については以下を参照してください。

6.1 Bytes、String、Float、Int のデータに対するフィルター

Bytes (p. 23) 、String (p. 25) 、Float (p. 26) 、Int (p. 27) （これらは文字列 ID を持つプリミティブデータです）に対するフィルター条件の設定方法は、以下の 2 種があります

個別フィルター

「指定されたチャンネルかつ指定されたデータタイプ」のデータについて、データ ID が完全一致する場合にだけ通過させるフィルター条件です。（例:「チャンネル 1 の Float データについて、データ ID が「ABC」であれば通過させる」）

All Pass フィルター

「指定されたチャンネルかつ指定されたデータタイプ」のデータについて、すべて通過させるフィルター条件です。（例:「チャンネル 1 の Float であれば通過させる」）

> **注釈:** 個別フィルター、All Pass フィルターともに、概念の説明のための一時的な名称です。実装においては、名称が異なる可能性がありますので、ご注意ください。

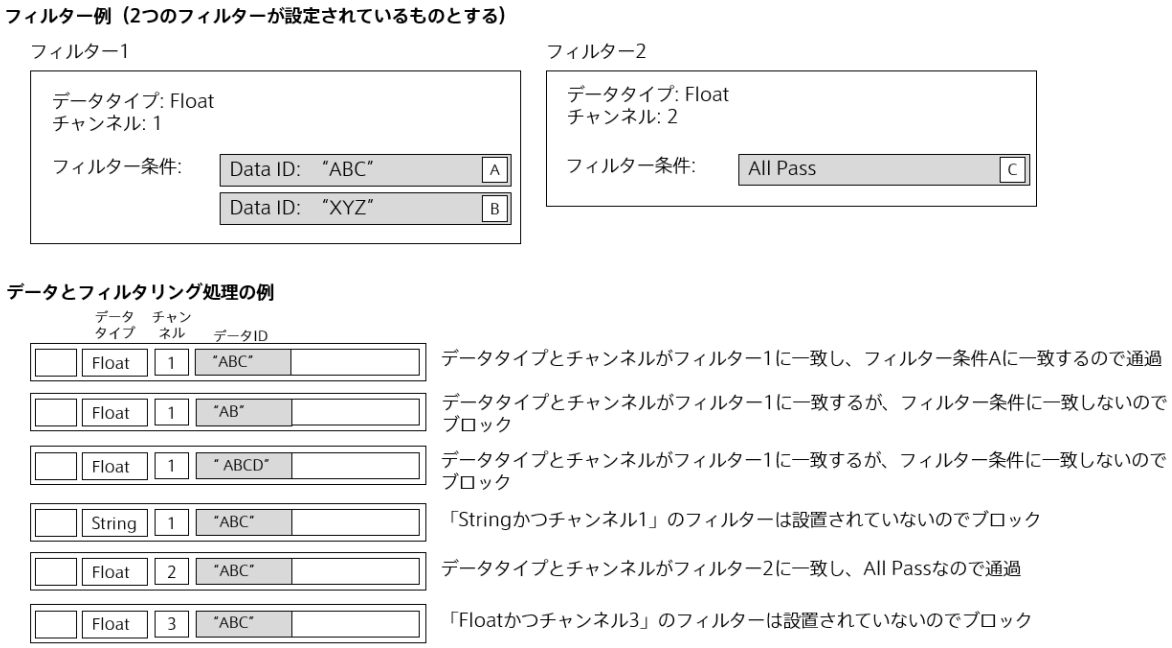


図 21　データ ID によるフィルタリング処理の例

6.2 CAN、General Sensor、NMEA、Control Pad、Generic のデータに対するフィルター

CAN (p. 32) 、General Sensor (p. 36) 、NMEA (p. 34) 、Controlpad (p. 42) 、Generic (p. 43) (数値 ID を持つデータ) に対するフィルター条件は、ID マスク、ID マスク結果、受入可否の 3 つのパラメーターにより設定します。

データ ID に「ID マスク」を適用し（ビットマスク、2 進数での論理積)、その結果が、指定された「ID マスク結果」と一致するならば、そのデータを通過させます（または、そのデータを拒否します)。

ID マスク

データ ID に適用するマスク値を指定します。

ID マスク結果

データ ID にマスクを適用した後の値がこの値と一致した場合に、そのデータをフィルター適用対象とします。

受入可否

フィルター適用対象となったデータを通過させるかを設定します。

例えば、「ID マスク : 0x0000FFFF」、「ID マスク結果 : 0x00000D3F」、「受け入れ : True」というフィルター条件は、「データ ID にマスク 0x0000FFFF を適用した結果が 0x00000D3F の場合は、そのデータを受け入れる」という意味です。

あるデータのデータ ID が「0xFFAB0D3F」のとき、「ID マスク : 0x0000FFFF」を適用すると、「0x00000D3F」となります。この値は、フィルターパラメーター「マスク結果 : 0x00000D3F」と一致します。「受け入れ True」が指定されていますので、このデータは受信することになります。

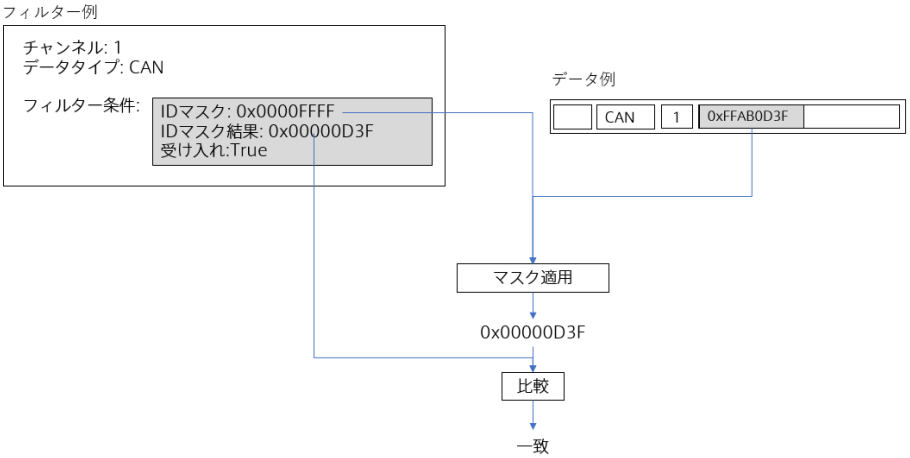
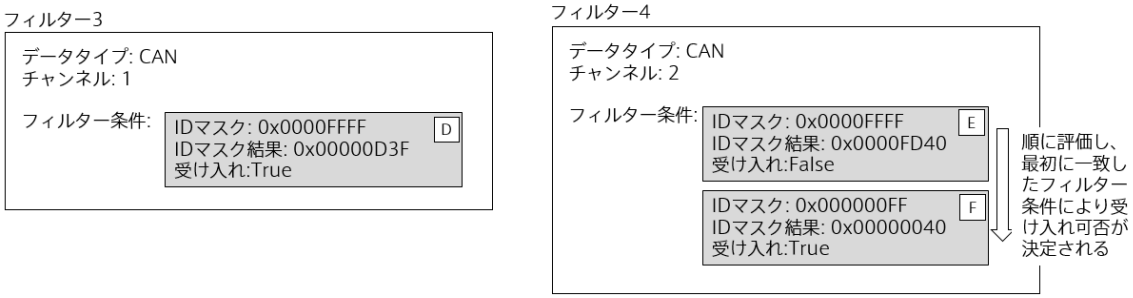


図 22　データ ID の一致判定

フィルター例（2つのフィルターが設定されているものとする）



データとフィルタリング処理の例

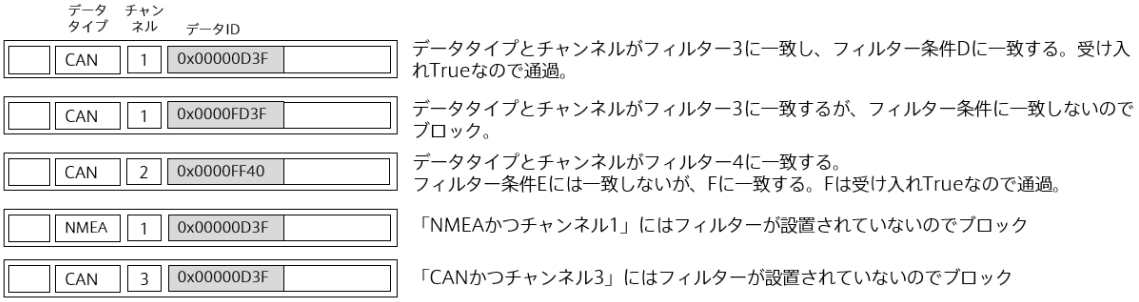


図 23　数値 ID によるフィルタリング処理の例