GARMIN.

Developer Guide for IPC Inbound

V 3.1.1

Updated: 10/10/2024

Contents

Revision History 4
IPC Inbound API 5
  Configuring IPC Inbound 6
  Costs 6
  Authentication 6
  Dates 6
  Altitudes 7
  Speeds 7
  Course 7
IPC Inbound API Documentation 7
HTTP Status Codes 8
Errors 8
  Error Codes 8
  JSON Error Object 9
    Example Error Result 10
Device Limits 10
  Versions 1.x 10
IPCv2 Endpoint Description 10
  /Emergency 10
    GET /Respondent 10
    GET /State?IMEI=tenant-imei1,tenant-imei2,… 11
    POST /AcknowledgeDeclareEmergency?IMEI=tenant-imei 11
    POST /SendMessage 11
  /Location 12
    GET /LocationRequest?IMEI=tenant-imei1,tenant-imei2,… 12
    GET /LastKnownLocation?IMEI=tenant-imei1,tenant-imei2,… 12
    POST /SendLocationRequest 12
    GET /History?IMEI=tenant-imei1,tenant-imei2&Start=2024-01-01&End=2024-03-01 12
  /Message 12
    POST /Message 12
    POST /Binary 14
    POST /Media 15

/Tracking 16
POST /Tracking 16
POST /Interval 17
Reference 18

Revision History

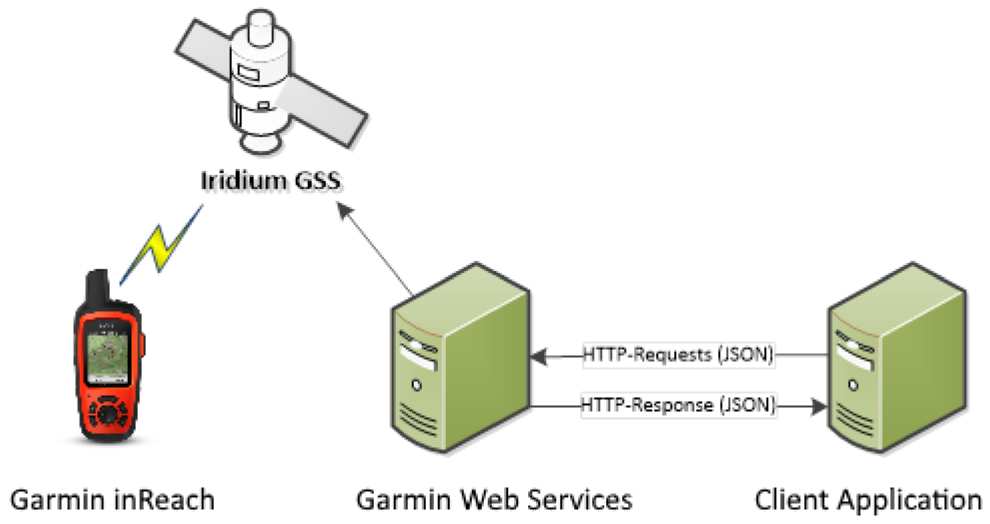
Date	Version	Description	Author
2012-03-12	1.0.1	Initial Draft Version	SWN
2012-03-13	1.0.2	Added Revision History	SWN
2012-03-13	1.0.3	Updated Emergency.svc	SWN
2012-03-29	1.0.4	Significant updates	SWN
2012-04-03	1.0.5	Review round with team	SWN
2012-04-05	1.0.6	Added label to the Location object and related error codes, fixed Tracking example	SWN
2012-04-23	1.0.7	Edited	BG
2012-05-21	1.0.8	Fixed example URLs	EBS
2012-05-10	1.09	Fixed example URL for LastKnownLocation	EBS
2013-01-07	1.1	Added Binary Message Correct Spelling Error Corrected Device Tracking Limit	EBS
2013-05-08	1.2	Added Emergency Service	EBS
2013-06-06	1.3	Added Pingback Service	AJA
2013-09-17	1.4	Made corrections to login URL	AJA
2014-03-26	1.5	Added Location History command	NDD
2014-03-28	1.6	Fixed erroneous labeling of ‘ReferencePoint’ field as ‘Location’ in Message object.	NDD
2014-07-30	1.7	Added Generic Binary	MDG
2014-10-21	1.8	Added EncryptedPinpoint	MDG
2014-11-14	1.9	Added wcf help page description	DMH
2015-01-07	1.10	Correct IPC link	MDG
2018-12-18	2.00	Garminized and Validated	
2019-04-01	2.01	Correct IPC link	
2021-11-17	2.02	Modernized links and documentation section	SRN
2024-04-03	3.0.0	Updated documentation to new IPC Inbound in IPC Service	RZ
2024-08-12	3.1.0	Added IPCv2 endpoint descriptions	RZ
2024-10-10	3.1.1	Added media encoding settings	RZ

IPC Inbound API

The inReach Portal Connect (IPC) Inbound API provides enterprise level inReach customers the ability to remotely control and communicate with an inReach device, manage tracking settings, send location requests and send text messages to their inReach users. The API is available through a collection of web services that receive requests and return responses in JSON.

The inReach device understands the following commands sent to it via the Iridium network. These commands are queued to be sent to a device via POST requests to the IPC web services.

- **Message Command**
  Sends a text message to the device containing a text field, sender, and optional location.
- **Locate Command**
  Requests the device report its current location.
- **Track On/Off Command**
  Turns on and off the automatic reporting of the device's location.
- **Track Interval Command**
  Changes the time between automatic reports of the device's location while tracking is enabled.



The IPC inbound API is currently comprised of seven JSON web services that can be located at the following URLs. Note that these are subject to change in the future.

https://{IPCInboundBaseUrl}/IPCInbound/V1/Messaging.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/Location.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/Tracking.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/Emergency.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/Pingback.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/AFF.svc
https://{IPCInboundBaseUrl}/IPCInbound/V1/Configuration.svc

To locate your tenant's IPC inbound API URL visit https://explore.garmin.com/IPC/, log in as a tenant admin, and then look for "Inbound URL" under "Inbound Settings". The URL found here can be substituted for "{IPCInboundBaseURL}" into the URLs above to form the full inbound API URL for you tenant.

For example, if your tenant’s IPC inbound API URL is “enterprise.inreach.garmin.com”, then the full Messaging.svc URL would be https://enterprise.inreach.garmin.com/IPCInbound/V1/Messaging.svc.

When using the API, it is important to know any limits on acceptable data ranges imposed by the destination unit. Please refer to the Device Limits section for further information.

Configuring IPC Inbound

To use API the tenant’s account must be correctly configured and an API username and password created.

- Log into https://explore.garmin.com/
- Click **Admin Controls.**
- Click on **Portal Connect.**
- Toggle the **Inbound Settings** slider to on.
- Enter a username and password and click **Save**.

Costs

Every command sent to a device from the Iridium GSS will incur a small charge. Generally, the cost associated with each one is dependent on the command sent and the size of the command contents. Check with your service plan for price per byte and included bytes.

The web services GET requests are used to query data at explore.garmin.com; no commands are sent to a device through the Iridium GSS and no costs are incurred. The POST request methods generally result in commands being sent to one or more devices and, potentially, messages returned from the device. The device owner’s plan will be charged according to the type and size of all commands sent, as well as message and tracks returned.

Authentication

The IPC API uses API key authentication to gain access to the web service. The API key is generated from the Explore application:

- Log into https://explore.garmin.com
- Click **Admin Controls**
- Click on **Portal Connect**
- On the **Inbound Settings** section, click **Generate API Key**
- Optionally configure an expiry date for the new API key

You can have up to three IPC Inbound API keys at the same time, with custom expiry dates. By default, the expiry date is 1 year. Set the expiry date for the API keys such as they give you time to change them continually without losing access to the API. The generated API key is then sent to the API with each request in the X-API-Key header. For example, if the API Key is “aab”, each IPC Inbound API request should have the X-API-Key: aab header. See https://en.wikipedia.org/wiki/API_key for a quick summary of the API key authentication.

Dates

All dates and timestamps in UTC are passed as JSON dates with the following format.
"\/Date(1333057093445)\/"

A date object stores a signed millisecond count with zero representing 1970-01-01 00:00:00 UTC.

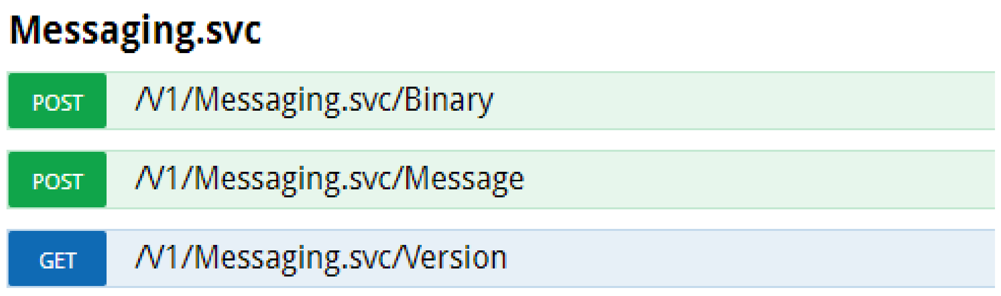
Altitudes
All altitudes are specified as heights in meters above the WGS 84 ellipsoid. The API supports heights between -1,000 and +18,000 meters inclusive. Refer to Device Limits for the limits and precision available for your device.

Speeds
Speeds are specified as positive numbers with km/h for the units. The API supports speeds from 0 to 1,854 km/h inclusive. Refer to Device Limits for the limits and precision available for your device.

Course
Course is specified as a positive or negative degree offset from true north. The range is -360° to +360° inclusive. Refer to Device Limits for the precision available for your device.

IPC Inbound API Documentation
More detailed IPC inbound API documentation can be found at https://explore.garmin.com/IPCInbound/docs. This documentation will always be current as it is automatically generated during the development of the API. Logging in is not required to access the documentation. For IPC v2, the documentation can be found here: IPC Inbound API (inreachapp.com)

For example, to find documentation for the messaging service, browse to https://explore.garmin.com/IPCInbound/docs and then expand the "Messaging.svc" section by clicking on it. For IPC v2, browse to IPC Inbound API (inreachapp.com) and expand the Messaging section.


Messaging.svc

POST /V1/Messaging.svc/Binary

POST /V1/Messaging.svc/Message

GET /V1/Messaging.svc/Version

When expanded, the documentation for a service will present a list of service operations. Clicking on individual service operations will expand them to reveal detailed explanations of the operation and technical details on how to interact with said operation programmatically.

The **MediaMessage** endpoint allows you to send media messages to a selected device. *This endpoint is not yet available and is subject to change. It will receive the following parameters, just like the normal **Message** endpoint:*

```
{ "Messages": [{
  "Recipients": ["integer(64)"],
  "Timestamp": "Date",
  "Payload": "Base64 encoded binary payload" }]}
```

HTTP Status Codes

The service operations provided by the IPC inbound API return standard HTTP status codes to reflect the success or failure of the requested action. The API will generally return one of the following status codes:

- **200 Success**
  This status will be returned by all API calls if completed successfully. The JSON object contained within is dependent on the API call.
- **401 Unauthorized**
  This status will be returned if the username and password are not present. The JSON object contained will be an error object.
- **403 Forbidden**
  This status will be returned if the username and password are incorrect. The JSON object contained will be an error object.
- **422 Unprocessable Entity**
  This status will be returned when the command is well-formed but cannot be executed due to a semantic error. The JSON object contained will be an error object.
- **429 Too Many Requests**
  This status will be returned when too many requests have been made to the IPC inbound service. To ensure the web services function satisfactorily under heavy demand, the services will automatically throttle disproportionate usage. Generally waiting a few seconds and trying again should resolve the issue.
  The response will contain the Retry-After header which will contain the number of seconds the client should wait until it tries again. The JSON object contained will be an error object.
- **500 Internal Error**
  This status will be returned when there is an internal failure in the inReach API. The JSON object contained will be an error object.
- **501 Not Implemented**
  This status will be returned by any unimplemented API function. The JSON object contained will be an error object.

Errors

When a web service request fails, it will return a JSON error object detailing the error.

Error Codes

The following error codes are defined. Future error codes will be appended to the end of this list.

Code	Name	Description
1	InternalError	An unexpected error occurred. This will be returned when the error does not fall into any other category.
2	TooManyRequestsError	Too many concurrent requests are being processed. This error will be returned when the server is under heavy usage and cannot satisfy your request. Waiting a few seconds and trying again should generally resolve the issue. The web services automatically throttle usage from all customers to ensure that a single customer does not saturate the server's capacity and negatively affect the performance for all.

3	AuthenticationError	Invalid username or password. Either the basic access authentication header is missing, or the passed credentials are invalid.
4	UnknownDeviceError	The specified IMEI does not belong to the tenant. Only devices belonging to the tenant can be used by the API.
5	InvalidMessageError	The message length is invalid. A text message may not be empty and has a maximum possible length of 160. When sending a reference point with a non-empty label, the length of the location's label counts towards this limit.
6	InvalidTimestampError	The message timestamp is invalid. The timestamp of the message must be on or after Jan 1, 2011, and cannot be in the future.
7	InvalidSenderError	The message sender is invalid. The sender must be a valid phone number or email address.
8	InvalidAltitudeError	The location's altitude is invalid. The height above the ellipsoid expressed in meters and must be between -1,000 and +18,000 meters inclusive.
9	InvalidSpeedError	The location's speed is invalid. The speed is expressed in km/h and must be between 0 and 1,854km/h inclusive.
10	InvalidCourseError	The location's course is invalid. The course must be between -360° and + 360° inclusive.
11	InvalidPositionError	The location's position is invalid. The latitude must be between -90° and +90° and the longitude between -180° and +180°.
12	InvalidIntervalError	The tracking interval is invalid. The tracking interval is specified in seconds and must be between 30 and 65535 seconds.
13	InvalidLocationTypeError	The location's type is invalid. The location type must be **0** (reference point) or **1** (GPS location).
14	InvalidLabelError	The location's label is invalid. The optional label may only be supplied when a reference point is sent, and its length is limited to 160 characters minus the length of the message.
15	IllegalEmergencyActionError	The account's emergencies are not handled by the account owner.
16	InvalidBinaryType	The binary type is invalid. The binary type must be **0** (Encrypted Binary), 1 (Generic Binary), or 2 (Encrypted Pinpoint).
17	InvalidPayloadError	The binary payload is invalid. The binary payload must be base64 encoded and no greater than 268 bytes.

JSON Error Object

Upon error the following error object will be returned

Name	Type	Description
Code	Number	An error code from Error Codes.
Description	String	An optional description of the error.
Message	String	A textual representation of the error code.
URL	String	The URL of the web request that failed.
IMEI	Array	An array of IMEIs that the request was unable to act upon and caused the request to fail.

Example Error Result

```
{
 "Code": 4,
 "Description": "The IMEI 300234010000000 is invalid.",
 "Message": "The specified IMEI does not belong to the tenant",
 "URL": "https:\/\/explore.garmin.com\/IPCInbound\/V1\/Messaging.svc",
 "Recipients": [300234010000000],
}
```

Device Limits

Versions 1.x

DeLorme inReach 1.x devices have the following limits

Altitude	-422 meters to +8,848 meters in increments of approximately 1 meter
Speed	0 km/h to 323 km/h in increments from 1km/h to 8km/h
Tracking Interval	Tracking intervals between 30 and 65535 seconds in 1 second increments
Course	The resolution of the course is limited to the 16 cardinal directions which are in increments of 22.5°.

IPCv2 Endpoints Description

/Emergency

GET /Respondent

Gets the status of the GEOSEnabled flag. The GEOSEnabled flag shows whether an SOS request is handled by the Garmin Response Team, or by a third-party search and rescue service. The call Returns a JSON string:

```
{
  “respondent”: 1
}
```

If GEOSEnabled is true, the respondent property will be 1, otherwise it will be 0. If the IPC Inbound API key is missing, the endpoint will return the status code 401 Unauthorized, with the following JSON string:

```
{
  “Code”: 3,
  “Message”: “Invalid api key”,
  “Description”: “Invalid api key”,
  “URL”: https://ipcinbound.inreachapp.com/api/Emergency/Respondent,
  “IMEI”: null
}
```

If the IPC Inbound API key is invalid, the endpoint will return the status code 403 Forbidden, with the following JSON string:

```
{
  “Code”: 3,
  “Message”: “Invalid api key”,
  “Description”: “Invalid api key”,
  “URL”: https://ipcinbound.inreachapp.com/api/Emergency/Respondent,
  “IMEI”: null
}
```

GET /State?IMEI=tenant-imei1,tenant-imei2,...

Gets the emergency state of the given IMEI. The IMEI parameter can be a single IMEI string, or multiple IMEIs separated by comma. The call returns a JSON string:

If one of the IMEIs is invalid or is not owned by the tenant, the endpoint will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 4,
    "Message": "Unknown or invalid device IMEI",
    "Description": "The IMEI 301434030944441 is invalid",
    "URL": "https://ipcinbound.inreachapp.com/api/Emergency/State?IMEI=...",
    "IMEI": [ … ]
}
```

POST /AcknowledgeDeclareEmergency?IMEI=tenant-imei

Acknowledges an SOS triggered by the given IMEI. The tenant must have the GEOSEnabled flag set to true. The call returns an empty body, with status code 200 OK if it was able to mark the SOS as acknowledged.

If the GEOSEnabled flag is false, the endpoint will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 15,
    "Message": "Illegal emergency action",
    "Description": "Illegal emergency action. Service support is provided by GEOS",
    "URL":
"https://ipcinbound.inreachapp.com/api/Emergency/AcknowledgeDeclareEmergency?IMEI=..."
,
    "IMEI": [ … ]
}
```

If the tenant doesn’t own the device with the given IMEI, the endpoint will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 4,
    "Message": "Unknown or invalid device IMEI",
    "Description": "The IMEI 301434030944441 is invalid",
    "URL":
"https://ipcinbound.inreachapp.com/api/Emergency/AcknowledgeDeclareEmergency?IMEI=..."
,
    "IMEI": [ … ]
}
```

POST /SendMessage

Requires a JSON body with the following properties:

```
{
  "IMEI": "tenant1-imei",
  "UtcTimeStamp": "2024-04-07T21:28:24.968Z",
  "Message": "This is a test from the new inbound service"
}
```

Sends the given message to the provided IMEI, with the provided time stamp. The call returns an empty body, with the status code 200 OK if it was able to send the message.

If the timestamp is in the future, the endpoint will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 6,
    "Message": "The message timestamp is invalid",
    "Description": "The message time 4/7/2066 9:28:24 PM is invalid, it may not be in
the future.",
    "URL": "https://ipcinbound.inreachapp.com/api/Emergency/SendMessage",
    "IMEI": null
}
```

If the IMEI is invalid or not owned by the tenant, the endpoint will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 4,
    "Message": "Unknown or invalid device IMEI",
    "Description": "The IMEI … is invalid",
    "URL": "https://ipcinbound.inreachapp.com/api/Emergency/SendMessage",
    "IMEI": [ … ]
}
```

/Location

GET /LocationRequest?IMEI=tenant-imei1,tenant-imei2,...

GET /LastKnownLocation?IMEI=tenant-imei1,tenant-imei2,...

POST /SendLocationRequest

Requires a JSON body with the following property:

```
{
  "IMEI": [ … ]
}
```

GET /History?IMEI=tenant-imei1,tenant-imei2&Start=2024-01-01&End=2024-03-01

/Message

POST /Message

The endpoint requires a JSON body with the following properties:

```
{
  "Messages": [{
      "Recipients": [ tenant-imei1, tenant-imei2, … ],
      "Sender": "test.ipc@gmail.com",
      "Timestamp": "\/Date(1296557000000)\/",
      "Message": "Hello is it me you're looking for?"}
  ]
}
```

We can also send a reference point with the message, by using the following JSON string:

```
{
  "Messages": [
    {
      "Recipients": [ “tenant-imei1”, … ],
      "Sender": "test.ipc@gmail.com",
      "Timestamp": "\/Date(1296557000000)\/",
      "Message": "New IPC Reference Point 1",
      "ReferencePoint": {
        "LocationType": "1",
        "Altitude": "20",
        "Speed": "20",
        "Course": "180",
        "Coordinate": {
          "Latitude": 2,
          "Longitude": 2
        },
        "Label": ""
      }
    }
  ]
}
```

The call returns 200 OK when sending the message, with the following JSON string:

```
{
    "count": 1
}
```

The count property specifies how many messages were sent, if multiple recipients and messages were specified.

The LocationType property must have the value 0 for Reference Point or 1 for GPS Location. If we send an invalid LocationType , we will get a 422 Unprocessable Entity erLocationType, the following JSON string:

```
{
    "Code": 13,
    "Message": "The location's type is invalid",
    "Description": "The LocationType 2 is invalid, it must be 0 or 1.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Message",
    "IMEI": [ tenant-imei1 ]
}
```

If we send an invalid altitude, we will get a 422 Unprocessable Entity error code, with the following JSON string:

```
{
    "Code": 8,
    "Message": "The location's altitude is invalid",
    "Description": "The altitude -1001 is invalid, it must be between -1000 and 
18000.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Message",
    "IMEI": [ tenant-imei1 ]
}
```

If we send an invalid speed, we will get a 422 Unprocessable Entity error code, with the following JSON string:

```
{
    "Code": 9,
    "Message": "The location's speed is invalid",
    "Description": "The speed -1 is invalid, it must be between 0 and 1854 km/h.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Message",
    "IMEI": [ tenant-imei1 ]
}
```

If we send an invalid course, we will get a 422 Unprocessable Entity error code, with the following JSON string:

```
{
    "Code": 10,
    "Message": "The location's course is invalid",
    "Description": "The course -361 is invalid, it must be between -360 and 360 
degrees.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Message",
    "IMEI": [ tenant-imei1 ]
}
```

The same will happen for invalid latitude and longitude. In case of an invalid API key, the call will return 401 Unauthorized, with the following JSON string:

```
{
    "Code": 3,
    "Message": "Invalid api key",
    "Description": "Invalid api key",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Message",
    "IMEI": null
}
```

POST /Binary

The endpoint requires a JSON body with the following properties:

```
{
  "Messages": [{
      "Recipients": [ "tenant-imei1", … ],
      "Type": 0,
      "Payload": "testbinaryfromnewipc" }
  ]
}
```

Type 0 is for sending encrypted messages, and type 1 for generic binary messages. The call will return 200 OK if the messages are sent, with the following JSON string:

```
{
    "count": 1
}
```

The count is the number of messages sent, according to the number of IMEIs provided and number of messages added to the message array.

If the message type is not 0 or 1, the call returns 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 16,
    "Message": "The message binary type is invalid",
    "Description": "The binary type is invalid.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Binary",
    "IMEI": [ tenant-imei1 ]
}
```

If the payload is invalid, for example contains invalid Base64 characters, the call will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 17,
    "Message": "The message payload is invalid",
    "Description": "The payload contains invalid Base64 characters.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Binary",
    "IMEI": [ tenant-imei1 ]
}
```

If the API key is invalid, the call will return 401 Unauthorized with the following JSON string:

```
{
    "Code": 3,
    "Message": "Invalid api key",
    "Description": "Invalid api key",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Binary",
    "IMEI": null
}
```

POST /Media

The endpoint requires a JSON body with the following properties:

```
{
  "Recipients": "tenant-imei1,tenant-imei2,…",
  "Sender": "test.media@garmin.com",
  "Message": "Test Media Message",
  "Timestamp": "2024-08-05T10:44:24.241Z",
  "Media": "data:image/avif;base64,imageBase64"
}
```

The call will return 200 OK if the messages are sent, with the following JSON string:

```
{
    "count": 1
}
```

If the payload is invalid, for example contains invalid Base64 characters, the call will return 422 Unprocessable Entity with the following JSON string:

```
{
    "Code": 17,
    "Message": "The message payload is invalid",
    "Description": "The payload contains invalid Base64 characters.",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Binary",
    "IMEI": [ tenant-imei1 ]
}
```

If the API key is invalid, the call will return 401 Unauthorized with the following JSON string:

```
{
    "Code": 3,
    "Message": "Invalid api key",
    "Description": "Invalid api key",
    "URL": "https://ipcinbound.inreachapp.com/api/Messaging/Binary",
    "IMEI": null
}
```

/Tracking

POST /Tracking

The call requires a JSON body with the following properties:

```
{
  "Devices": [{
      "Imei": tenant-imei,
      "Tracking": false,
      "Interval": 30
    }
  ]
}
```

Using the Tracking property, we can turn off tracking or on, and we can set the tracking interval using the Interval property. The call will return 200 OK if successful.

If we send tracking commands to a device that is currently in emergency mode, the call will return 500 Internal Server Error with the following JSON string:

```
{
    "Code": 1,
    "Message": "An unexpected error occurred",
    "Description": "Tracking commands cannot be sent to a device in SOS.",
    "URL": "https://ipcinbound.inreachapp.com/api/Tracking/Tracking",
    "IMEI": [ … ]
}
```

For an invalid IMEI, the call returns 422 Unprocessable Entity, with the following JSON string:

```
{
    "Code": 4,
    "Message": "Unknown or invalid device IMEI",
    "Description": "The IMEI … is invalid",
    "URL": "https://ipcinbound.inreachapp.com/api/Tracking/Tracking",
    "IMEI": [ … ]
}
```

The call will also return 422 Unprocessable Entity if the tracking interval lower or upper limit is invalid, or if the IMEI does not belong to the tenant.

POST /Interval

This call requires a JSON body with the following properties:

```
{
  "Devices": [{
      "Imei": tenant-imei,
      "Interval": 600
    }
  ]
}
```

Using the Interval property, we can configure the tracking interval for the device. The call will return 200 OK if the command was successful.

The call will also return 422 Unprocessable Entity if the tracking interval lower or upper limit is invalid, or if the IMEI does not belong to the tenant, with a JSON string similar to this:

```
{
    "Code": 4,
    "Message": "Unknown or invalid device IMEI",
    "Description": "The IMEI … is invalid",
    "URL": "https://ipcinbound.inreachapp.com/api/Tracking/Interval",
    "IMEI": [ … ]
}
```

IPC v2 media message specification

We can send media messages using the messaging endpoints. The accepted formats are:

- For audio: MPEG, OGG, WEBM
- For photo: JPEG, PNG, AVIF

Internally, audio files are encoded as OGG 8000 Hz, VBR, mono.

Reference

The latest IPC Documentation is available online at the following locations.

IPC Inbound: https://developer.garmin.com/inReach/IPC_Inbound.pdf

IPC Outbound: https://developer.garmin.com/inReach/IPC_Outbound.pdf