WELCOME TO THE **DEVELOPER PORTAL**

Spire Sense API Documentation

Getting Started

Examples

Quick Reference
Vessels API
Messages API
TCP Feed

Support

Spire Answers
System Status
Chat & Email

GETTING STARTED

AUTHENTICATION

Our APIs use tokens to authenticate requests. Attempting to make requests to the API without a valid JWT will result in the return of an HTTP 401 Not Authorized response code containing a WWW-Authenticate HTTP header with an error message.

In addition, to ensure transport layer security, all access or communication with the API must be made over HTTPS.

Host	`ais.spire.com`
Header Key	Authorization
Header Value	Bearer {your_token}

RESPONSE FORMAT

Responses from an API request are formatted as JSON. All responses have a common schema as this example response from our Vessels API shows:

```
{
    "paging": {
        "limit": 100,
        "total": 1394,
        "next": "Mg=="
    },
            "mmsi": 538006090,
            "imo": 9700665,
            "call_sign": "V7JP9",
            "ship_type": "Cargo",
            "class": "A",
            "flag": "MH",
            "length": 199,
            "width": 32,
            "ais_version": 0,
            "updated_at": "2017-12-19T00:11:48.202337+00:00",
            "last_known_position": {
                "timestamp": "2017-12-18T20:24:27+00:00",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        60.87363,
                        -13.02203
                    ]
                },
                "heading": 242,
                "speed": 12.1,
                "rot": 4,
                "accuracy": null,
                "collection_type": "satellite",
                "draught": null,
                "maneuver": 0,
                "course": 247.1
            },
            "most_recent_voyage": {
                "eta": "2017-12-26T14:00:00+00:00",
                "destination": "DURCAN           0"
            },
            "predicted_position": {
                "timestamp": "2017-12-19T01:36:07+00:00",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        59.882200289,
                        -13.4274815048
                    ]
                },
                "speed": 12.1,
                "course": 247.1,
                "confidence_radius": 38.3607
            },
            "general_classification": "All Other Activities",
            "individual_classification": "Bulk Carrier",
            "gross_tonnage": "36286",
            "lifeboats": null,
            "person_capacity": null
        }
    ]
}
```

Field	Description

`data` *json*	The data returned from the request in the form of an array.

DATA TYPES

There are a few basic data types found throughout our APIs. Data types are used to describe resource properties returned from the API. They are also used to specify input into our systems as filter parameters.

Data Type	Description
`string`	String value.
`date`	Dates conforming to ISO 8601.
`integer`	Integer value.
`number`	Numeric value with variable precision; includes floats, decimals.
`geometry`	Input and response geometry as GeoJSON.
`bool`	Having one of two possible values of true or false.
`array`	JSON array.
`json`	JSON object.

Geometry

Geographic features returned by our APIs are represented as GeoJSON objects. Geometries can also be used to spatially filter your queries. Responses return points while inputs accept polygons and multipolygons.

Dates

Date and time responses and filters conform to the ISO 8601 standard with time represented in the UTC timezone. The generic ISO 8601 timestamp representation is: YYYY-MM-DDTHH:MM:SSZ.

```
"timestamp": "2017-03-07T22:05:33"
```

variety of filter types for our APIs. Below is a list of common types with the actual filters listed under Filter Parameters section of each API.

Exact Match

Returns records where there is a specific match for the value provided. For example:

```
https://ais.spire.com/messages?collection_type=satellite
```

List

Returns results that match multiple values. Any field that support list also supports exact match. For example:

```
https://ais.spire.com/messages?mmsi=239245000,273216800,249810000
```

Range

Range filters work on some fields that are dates and strings. For example:

```
https://ais.spire.com/messages?received_after=2017-03-01T16:56:15&received_before=201
01T17:56:15
```

Geospatial

Returns data that has a geospatial intersection with the provided input geometry. Input geometries should be valid GeoJSON polygons and multipolygons. For example:

```
https://ais.spire.com/messages?position={"type":"Polygon","coordinates":
[[[-122.41269350051881,37.76058796575955],[-122.41269350051881,37.764124860544094]
[-122.40750074386597,37.764124860544094],[-122.40750074386597,37.76058796575955],
[-122.41269350051881,37.76058796575955]]]}
```

PAGINATION

We have implemented two different types of pagination: cursors for more easily working with constantly updating data feeds and limit & offsets for working with more static data sets.

Cursors

well.

By default, the Messages API returns results for the past three hours. But if you make the same query 30 minutes later, you will not return the same results as new messages have been added while others have dropped out because they are no longer in the three-hour window.

What most customers want to do is get all the data from the feed since the last time they queried the API. To make this easier we have introduced the `since` cursor so that you can quickly request new data from where you were before.

Example First Request

First Messages API request to get data for the past three hours:

```
GET https://ais.spire.com/messages?fields=decoded
```

Example First Paging Response

This will return the first "page" of 20,000 results for the past three hours. A since cursor is provided in the body of the response that points to the end of the results that have been returned.

```
{
  "paging": {
    "since": "MTAxNTExOTQ1MjAwNzI5NDE=",
    "limit": "20000"
  }
}
```

Example Second Request

We then use this since cursor in the next request to the API. This will return the next "page" of results moving closer in time to now.

```
GET https://ais.spire.com/messages?fields=decoded&since=MTAxNTExOTQ1MjAwNzI5NDE=
```

Continue to cycle through the pages with each new cursor until you get a response with an empty array. You are now caught up with the feed. From there you can continue requesting on a regular cadence (ex: 5 minutes) to keep up with new data and move the cursor forward.

Limit & Offset

of results.

For example, to move to the next page of results, simply append the `next` or `after` query parameter:

```
https://ais.spire.com/vessels?next=Mg==
```

```
https://ais.spire.com/messages?
fields=decoded&after=MjAxNy0wOC0yOCAwMDowMDowMCswMDowMA==
```

To move to the previous page, append the `previous` or `before` query parameter:

```
https://ais.spire.com/vessels?previous=Mg==
```

```
https://ais.spire.com/messages?
fields=decoded&before=MjAxNy0wOC0yOCAwMDowMDowMCswMDowMA==
```

SORTING

You can specify your sort order by including `sort={PROPERTY_NAME}` in the query string. Use a minus sign (-) to denote descending sort order: `sort=-{PROPERTY_NAME}`.

By default, the Vessels API is sorted by `created_at` date/time in ascending order. It can also be sorted by `updated_at`.

```
https://ais.spire.com/vessels?sort=updated_at
```

By default, the Messages API is sorted by `timestamp` date/time in ascending order. It can also be sorted by `created_at`.

```
GET https://ais.spire.com/messages?sort=created_at
```

LIMITS

Requests made to our APIs may lead to thousands of available results. Because of this, when a request is made to the REST API, all of the results usually won't be received in a single response.

```
https://ais.spire.com/messages?limit=10
```

The Vessels API has a default limit of 100 with a max of 1000.

The Messages API has a default limit of 20000 with a max of 20000.

ERRORS

When there is an error with your request, the response header will contain a status code to help you determine what the issue is. Additionally, the response body will contain a more detailed message.

Our APIs may respond with the following errors:

400	Bad Request A request made with a malformed HTTP Authorization Header or query parameters. Unaccepted query parameters will simply be ignored.
401	Unauthorized A request made with an invalid, unrecognized or missing access token.
403	Forbidden The metadata associated to a JWT is no longer valid and access to the API is denied.
404	Not Found A request made to an unknown or supported resource.
406	Not Acceptable A request made with invalid HTTP headers.

	The request was well-formed but is too large.
422	Unprocessable The request was well-formed but was unable to be followed due to semantic errors.
429	Too Many Requests Exceeding the rate limit will result in a 429 error response until a rate limit refresh threshold has been met.
503	Service Unavailable If the API encounters any technical difficulties while processing a request, it will respond with a description detailing the status of the API.

EXAMPLES

POSTMAN COLLECTION

The fastest way to get started with the Spire API is to download our open source Postman collection. All you have to is drop in your access token and you will be making Vessels and Messages calls in seconds.

Spire API Postman Collection.

LIST ALL VESSELS

Request

```
{
    "paging": {
        "total": 261271,
        "limit": 100,
        "next": "Mg=="
    },
    "data": [
        {
            "individual_classification": "Bulk Carrier",
            "most_recent_position": {
                "maneuver": 0,
                "course": 279,
                "draught": 12,
                "timestamp": "2017-08-25T12:37:39+00:00",
                "rot": 0,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        -179.1028,
                        43.94133
                    ]
                },
                "collection_type": "satellite",
                "speed": 12.9,
                "heading": 276,
                "accuracy": null
            },
            "name": "NEW DIRECTION",
            "width": 32,
            "updated_at": "2017-08-29T21:15:25.331593+00:00",
            "person_capacity": null,
            "mmsi": 355800000,
            "ais_version": 0,
            "length": 190,
            "imo": 9622801,
            "id": "69b8b52c-5f72-45a4-8317-3133e9662f91",
            "gross_tonnage": "31768",
            "class": "A",
            "flag": "PA",
            "ship_type": "Cargo",
            "general_classification": "Merchant",
            "most_recent_voyage": {
                "destination": "NING BO",
                "eta": "2017-09-05T12:00:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "3FRY3"
        }
    ]
}
```

GET INDIVIDUAL VESSEL

```
curl -H "Authorization: Bearer {your_token}" -X GET
https://ais.spire.com/vessels/a5b738b4-faf0-4a7e-9a87-1c0ccfb123d2
```

Response

```
{
    "data": {
        "individual_classification": "Passenger Ship",
        "most_recent_position": {
            "maneuver": 0,
            "course": 35,
            "draught": 9.3,
            "timestamp": "2017-08-24T23:55:33+00:00",
            "rot": 0,
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -86.96772,
                    20.59382
                ]
            },
            "collection_type": "terrestrial",
            "speed": 18,
            "heading": 38,
            "accuracy": null
        },
        "name": "HARMONY OF THE SEAS",
        "width": 66,
        "updated_at": "2017-08-27T16:43:22.367784+00:00",
        "person_capacity": 8880,
        "mmsi": 311000396,
        "ais_version": 0,
        "length": 362,
        "imo": 9682875,
        "id": "a5b738b4-faf0-4a7e-9a87-1c0ccfb123d2",
        "gross_tonnage": "226963",
        "class": "A",
        "flag": "BS",
        "ship_type": "Passenger",
        "general_classification": "Merchant",
        "most_recent_voyage": {
            "destination": "PORT EVERGLADES",
            "eta": "2017-08-26T09:00:00+00:00"
        },
        "lifeboats": null,
        "call_sign": "C6BX8"
    }
}
```

LIST ALL CARGO VESSELS

Request

```
{
    "paging": {
        "total": 51387,
        "next": "Mg=="
    },
    "data": [
        {
            "individual_classification": "Cargo Ship",
            "most_recent_position": {
                "maneuver": 0,
                "course": 162.8,
                "draught": 6.9,
                "timestamp": "2017-08-25T11:16:14+00:00",
                "rot": 0,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        13.28397,
                        -22.97013
                    ]
                },
                "collection_type": "satellite",
                "speed": 12,
                "heading": 160,
                "accuracy": null
            },
            "name": "OCEAN GLOBE",
            "width": 23,
            "updated_at": "2017-08-29T21:15:25.333376+00:00",
            "person_capacity": null,
            "mmsi": 367649340,
            "ais_version": 0,
            "length": 166,
            "imo": 9419008,
            "id": "c504c803-5d9e-41a9-b4a4-2f11aaf5af73",
            "gross_tonnage": "15549.0",
            "class": "A",
            "flag": "US",
            "ship_type": "Cargo",
            "general_classification": "Merchant",
            "most_recent_voyage": {
                "destination": "DURBAN SA",
                "eta": "2017-08-31T06:00:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "KOGE"
        }
}
```

LIST ALL CHINESE CARGO VESSELS

Request

Response

```
{
    "paging": {
        "total": 24904,
        "limit": 100,
        "next": "Mg=="
    },
    "data": [
        {
            "individual_classification": "Bulk Carrier",
            "most_recent_position": {
                "maneuver": 0,
                "course": 302.7,
                "draught": 10.7,
                "timestamp": "2017-08-25T14:38:08+00:00",
                "rot": 0,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        120.83391,
                        27.99017
                    ]
                },
                "collection_type": "terrestrial",
                "speed": 0,
                "heading": 241,
                "accuracy": null
            },
            "name": "PU SHENG 8",
            "width": 24,
            "updated_at": "2017-08-29T20:28:25.609064+00:00",
            "person_capacity": null,
            "mmsi": 413440220,
            "ais_version": 0,
            "length": 173,
            "imo": null,
            "id": "c186473e-7a8d-4274-bf2a-7bfb7395e2a7",
            "gross_tonnage": "16539",
            "class": "A",
            "flag": "CN",
            "ship_type": "Cargo",
            "general_classification": "Merchant",
            "most_recent_voyage": {
                "destination": "WEN ZHOU",
                "eta": "2017-08-24T12:30:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "BKSZ6"
        }
```

LIST ALL TANKERS IN THE NORTH SEA

```
'https://ais.spire.com/vessels/' -d '?ship_type=tanker&geometry_within={
        "type": "Polygon",
        "coordinates": [
          [
            [
              -5.9765625,
              51.31688050404585
            ],
            [
              12.12890625,
              51.31688050404585
            ],
            [
              12.12890625,
              61.39671887310411
            ],
            [
              -5.9765625,
              61.39671887310411
            ],
            [
              -5.9765625,
              51.31688050404585
            ]
          ]
        ]
      }'
```

Response

```
{
    "paging": {
        "total": 1416,
        "limit": 100,
        "next": "Mg=="
    },
    "data": [
        {
            "individual_classification": "Buoy Ship",
            "most_recent_position": {
                "maneuver": 1,
                "course": 0,
                "draught": 6.7,
                "timestamp": "2017-08-25T14:46:12+00:00",
                "rot": 0,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        12.08969,
                        54.11197
                    ]
                },
                "collection_type": "terrestrial",
                "speed": 0,
                "heading": 253,
                "accuracy": null
            "person_capacity": null,
            "mmsi": 211638130,
            "ais_version": 0,
            "length": 168,
            "imo": 6818617,
            "id": "b20b3d61-5e6f-45f3-82e2-a9cd5119d109",
            "gross_tonnage": "4937.0",
            "class": "A",
            "flag": "DE",
            "ship_type": "Tanker",
            "general_classification": "Merchant",
            "most_recent_voyage": {
                "destination": "TALLINN_ANCHORAGE",
                "eta": "2017-08-27T15:00:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "DJFO2"
        }
    ]
}
```

FIND VESSELS BY MMSI LIST

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
https://ais.spire.com/vessels?mmsi=219002418,244710824,244780327,457545000
```

Response

```
{
    "paging": {
        "total": 4,
        "limit": 100
    },
    "data": [
        {
            "individual_classification": "Passenger Ship",
            "most_recent_position": {
                "maneuver": 0,
                "course": 47,
                "draught": null,
                "timestamp": "2017-08-24T08:17:40+00:00",
                "rot": 731,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        15.03868,
                        55.25095
                    ]
                },
                "collection_type": "terrestrial",
                "speed": 9.8,
            "width": 7,
            "updated_at": "2017-08-27T15:08:00.477696+00:00",
            "person_capacity": null,
            "mmsi": 219002418,
            "ais_version": 0,
            "length": 23,
            "imo": null,
            "id": "9eccf900-b252-4056-8740-e989e368f4e1",
            "gross_tonnage": "0",
            "class": "A",
            "flag": "DK",
            "ship_type": "Passenger",
            "general_classification": "Merchant",
            "most_recent_voyage": {
                "destination": null,
                "eta": null
            },
            "lifeboats": null,
            "call_sign": "OUQD"
        }
    ]
}
```

FIND VESSELS BY IMO LIST

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
https://ais.spire.com/vessels?imo=9799666,9363273,9180011
```

Response

```
{
    "paging": {
        "total": 3,
        "limit": 100,
    },
    "data": [
        {
            "individual_classification": null,
            "most_recent_position": {
                "maneuver": 0,
                "course": 167.9,
                "draught": 4.2,
                "timestamp": "2017-08-25T14:25:53+00:00",
                "rot": 1,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        103.80757,
                        1.24418
                    ]
                },
            },
            "name": "ASPIRE",
            "width": 17,
            "updated_at": "2017-08-29T20:52:26.288859+00:00",
            "person_capacity": null,
            "mmsi": 563006700,
            "ais_version": 0,
            "length": 97,
            "imo": 9799666,
            "id": "dbf8e47a-f937-4ced-9aca-807026a6c46a",
            "gross_tonnage": null,
            "class": "A",
            "flag": "SG",
            "ship_type": "Tanker",
            "general_classification": null,
            "most_recent_voyage": {
                "destination": "AWPA",
                "eta": "2017-08-23T12:30:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "9V5139"
        }
    ]
}
```

FIND VESSELS ARRIVING BETWEEN DATE/TIME

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
'https://ais.spire.com/vessels?arriving_after=2017-08-
29T03:00:00Z&arriving_before=2017-08-30T12:00:00Z'
```

Response

```
{
    "paging": {
        "total": 1165,
        "limit": 100,
        "next": "Mg=="
    },
    "data": [
        {
            "individual_classification": null,
            "most_recent_position": {
                "maneuver": 0,
                "course": 329.3,
                "draught": 10,
                "timestamp": "2017-08-25T04:56:05+00:00",
                "rot": 0,
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                },
                "collection_type": "satellite",
                "speed": 3.7,
                "heading": 339,
                "accuracy": null
            },
            "name": "TOMINI AMITY",
            "width": 30,
            "updated_at": "2017-08-29T19:53:24.797752+00:00",
            "person_capacity": null,
            "mmsi": 538007107,
            "ais_version": 0,
            "length": 186,
            "imo": 9109902,
            "id": "32018d11-fd57-41c4-aaad-a3afcc3f8a0b",
            "gross_tonnage": null,
            "class": "A",
            "flag": "MH",
            "ship_type": "Cargo",
            "general_classification": null,
            "most_recent_voyage": {
                "destination": "TEMA",
                "eta": "2017-08-30T02:00:00+00:00"
            },
            "lifeboats": null,
            "call_sign": "V7QY6"
        }
    ]
}
```

LIST MOST RECENT DECODED MESSAGES

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET https://ais.spire.com/messages
fields=decoded
```

Response

```
{
  "paging": {
    "limit": "20000",
    "since": "MjAxNy0wMy0yNCAxNzowNzoxNi4yMzUyNTc=",
    "actual": "20000+"
  },
  "data": [
    {
      "msg_type": 5,
      "msg_id": "1490375235_356446000",
      "collection_type": "terrestrial",
      "nmea": "!AIVDM,2,1,1,A,55Csg<82=RqdPu<n22118Tp<E=>0u04j2222221@:`G5=tt=0?
2T85Bh`888,0*7A\r\n!AIVDM,2,2,1,A,88888888880,2*25",
      "call_sign": "HOSM",
      "ais_version": 2,
      "type": 5,
      "draught": 6,
      "timestamp": "2017-03-24T17:07:15.828897+00:00",
      "mmsi": 356446000,
      "flag": "Panama",
      "ship_type": "Tanker",
      "name": "PRINCESS OPAL",
      "created_at": "2017-03-24T17:07:16.235257",
      "msg_description": "static",
      "length": 108,
      "eta": "2017-03-24T13:00:00",
      "flag_short_code": "PA"
    }
  ]
}
```

MOST RECENT MESSAGES MMSI LIST

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
https://ais.spire.com/messages?fields=decoded&mmsi=356206000,219657000,244992000
```

Response

```
{
  "paging": {
    "limit": "20000",
    "since": "MjAxNy0wMy0yNCAxNzoxMDozMy43NDQxOTY=",
    "actual": "79"
  },
  "data": [
    {
      "msg_type": 1,
      "msg_id": "1490375406_356206000",
      "course": 166.5,
      "collection_type": "terrestrial",
      "nmea": "!AIVDM,1,1,0,A,15Ce5d002:Q@1mFFGRBFPED<0>`<,0*0E",
      "rot": 0,
      "speed": 13.8,
      "latitude": 39.0904683333,
      "type": 1,
      "accuracy": 1,
      "status": 0,
      "maneuver": 0,
      "timestamp": "2017-03-24T17:10:06+00:00",
      "mmsi": 356206000,
      "flag": "Panama",
      "created_at": "2017-03-24T17:10:33.744196",
      "msg_description": "position",
      "longitude": 17.482525,
      "flag_short_code": "PA",
      "position": {
        ]
      },
      "heading": 170
    }
  ]
}
```

MOST RECENT TYPE 1 MESSAGES FOR AOI

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
'https://ais.spire.com/messages?msg_type=1&fields=msg_type,position&position=
{"type":"Polygon","coordinates":[[[-48.33984375,48.10743118848039],
[-71.015625,28.92163128242129],[-68.203125,24.686952411999155],
[-52.20703125,15.623036831528264],[-24.43359375,17.308687886770034],
[-13.7109375,47.754097979680026],[-48.33984375,48.10743118848039]]]}'
```

Response

```
{
  "paging": {
    "limit": "20000",
    "since": "MjAxNy0wMy0yNCAxNzoxMTowMy40NTg5NjM=",
    "actual": "20000+"
  },
  "data": [
    {
      "position": {
        "type": "Point",
        "coordinates": [
          -29.6648133333,
          38.3556583333
        ]
      },
      "mmsi": 245995000,
      "nmea": "!AIVDM,1,1,0,B,1CbVEv300<Mp=8hEt`4nJbN60@1N,0*6B",
      "msg_type": 1,
      "timestamp": "2017-03-24T17:11:03+00:00"
    }
  ]
}
```

ALL MESSAGES FROM DATE/TIME WINDOW

Request

```
curl -i -H "Authorization: Bearer {your_token}" -X GET
'https://ais.spire.com/messages?fields=decoded&received_after=2017-08-
```

```
{
    "paging": {
        "limit": "20000",
        "after": "MjAxNy0wOC0yOCAwMDowMDoyMiswMDowMA==",
        "actual": "20000+",
        "before": "MjAxNy0wOC0yOCAyMzo1OTo1OSswMDowMA=="
    },
    "data": [
        {
            "msg_type": 1,
            "msg_id": "1503878400_224335000",
            "course": 79.8,
            "collection_type": "satellite",
            "nmea": "!AIVDM,1,1,,B,13EtDV0P0I0J1F1iu`d37gv00000,0*1C",
            "rot": 731,
            "speed": 2.5,
            "source": "FM49",
            "latitude": -24.5313333333,
            "type": 1,
            "accuracy": 0,
            "status": 0,
            "maneuver": 0,
            "timestamp": "2017-08-28T00:00:00+00:00",
            "mmsi": 224335000,
            "flag": "Spain",
            "created_at": "2017-08-28T12:09:10.601135+00:00",
            "msg_description": "position",
            "longitude": 5.6843733333,
            "flag_short_code": "ES",
            "position": {
                "type": "Point",
                "coordinates": [
                    5.6843733333,
                    -24.5313333333
                ]
            },
            "heading": 511
        }
    ]
}
```

LIST ALL VESSELS PREDICTED OR LAST KNOWN IN THE INDIAN OCEAN

Request

```
curl -ig -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjdXN0b21lciI6eyJpZCI6IjM2In0sImlhdCI6MTQ3OTE
1NjUyNSwiaXNzIjoic3BpcmUifQ.kGMq4-U1hg_SAeauO8VmChcaZzB9CV8_y1ZHRa_pRm4" -X GET
'https://ais.spire.com/vessels/' -d '?ship_type=tanker&geometry_within={
        "type": "Polygon",
        "coordinates": [
            ],
            [
              12.12890625,
              51.31688050404585
            ],
            [
              12.12890625,
              61.39671887310411
            ],
            [
              -5.9765625,
              61.39671887310411
            ],
            [
              -5.9765625,
              51.31688050404585
            ]
          ]
        ]
      }'
```

Response

```
{
    "paging": {
        "limit": 100,
        "total": 1394,
        "next": "Mg=="
    },
    "data": [
        {
            "id": "92049dab-37cc-40a8-9196-8b001fcc2601",
            "name": "S TTH0OBT",
            "mmsi": 538007406,
            "imo": null,
            "call_sign": "V7FG5",
            "ship_type": "Other",
            "class": "A",
            "flag": "MH",
            "length": 199,
            "width": 32,
            "ais_version": 0,
            "updated_at": "2017-12-19T01:49:17.975182+00:00",
            "last_known_position": {
                "timestamp": "2017-12-19T00:48:57+00:00",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        96.50852,
                        -29.87481
                    ]
                },
                "heading": 270,
                "speed": 11.6,
                "rot": 0,
                "accuracy": null,
                "collection_type": "satellite",
            ,
            "most_recent_voyage": {
                "eta": "2017-12-28T03:00:00+00:00",
                "destination": "RICHARDS BAY"
            },
            "predicted_position": {
                "timestamp": "2017-12-19T01:51:06+00:00",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        96.2780232374,
                        -29.864833631
                    ]
                },
                "speed": 11.6,
                "course": 272.8,
                "confidence_radius": 6.0259
            },
            "general_classification": "All Other Activities",
            "individual_classification": "Bulk Carrier",
            "gross_tonnage": "35812",
            "lifeboats": null,
            "person_capacity": null
        }
    ]
}
```

CODE SAMPLES

Messages API Live Query

Python Java

```
"""
An example code to run live queries on the SPIRE API.
"""

import requests
import json
import time

# SPIRE AIS ENDPOINT
ENDPOINT = 'https://ais.spire.com/messages'

# FORMAT
FORMAT = 'json'

# YOUR TOKEN
AUTH_TOKEN =
"eyJhbGciOifaketokenJIUFAKEInR5cCI6IkpXVCJ9.eyJpYXQiOjE0NjcyMzg1MTgsImN1c3RvbWVyI

HEADERS = {"Authorization": "Bearer {}".format(AUTH_TOKEN), 'Accept': 'applicatio
```

```
def process_messages(messages):
    '''Function that will be used to process data fetched from the API'''
    print len(messages), 'messages'


def query_data():
    print 'Start Querying SPIRE Data...'
    request = ENDPOINT
    prev_since = None

    while True:
        print request
        response = requests.get(request, headers=HEADERS)

        data = response.json()

        try:
            process_messages(data['data'])
        except KeyError:
            print "No new data, move along..."
            continue

        if 'paging' in data:
            print data['paging']
            since = data['paging']['since']
            request = ENDPOINT + "?since=%s" % since
        else:
            print 'The data transfer is over. Thank you.'
            return

        if prev_since == since:
            print 'Waiting for 1 minute.'
            time.sleep(60)
        else:
            prev_since = since

if __name__ == '__main__':
    query_data()
```

Messages API Historical Query

Python Java

```
"""
An example code to run historical queries on the SPIRE API.
"""

import requests
import json

# SPIRE AIS ENDPOINT
ENDPOINT = 'https://ais.spire.com/messages'
```

```
AUTH_TOKEN =
"eyJhbGciOifaketokenJIUFAKEInR5cCI6IkpXVCJ9.eyJpYXQiOjE0NjcyMzg1MTgsImN1c3RvbWVyIW
As"

HEADERS = {"Authorization": "Bearer {}".format(AUTH_TOKEN), 'Accept':
'application/%s' % FORMAT}


# Message Processing
def process_messages(messages):
    '''Function that will be used to process data fetched from the API'''
    print
    messages


def query_data(received_after, received_before):
    request = ENDPOINT + "?received_after=%s&received_before=%s" %
(received_after, received_before)

    print
    'Start Querying SPIRE Data...'

    while True:

        response = requests.get(request, headers=HEADERS)

        data = response.json()

        process_messages(data['data'])

        if 'paging' in data:
            after = data['paging']['after']
            before = data['paging']['before']

            request = ENDPOINT + "?after=%s&before=%s" % (after, before)  #
(before, after) #

        else:
            print
            'The data transfer is over. Thank you.'
            return


if __name__ == '__main__':
    received_after = "2016-06-25T16:56:15"
    received_before = "2016-07-02T23:59:15"
    query_data(received_after, received_before)
```

TCP Socket Client

Python Java

```
"""
```

i

```
import time

server = "SPIRE SERVER"
port = SPIRE SERVER PORT

# YOUR TOKEN
AUTH_TOKEN =
"eyJhbGciOiJIUzI1NiIsInR5clalafaketokenCI6IkpXVCJ9.eyJpYXQiOjE0NjcyMzg1MTgsImN1VyI
jp7ImlkIjoiMTIifX0.D4GYo-LDASFoYLxUmScNUXO_YrYTNatWAs"


def connect():
    """
    A simple connection function to create a client socket.
    """
    global sock
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((server, port))
    login = 'A|T|' + AUTH_TOKEN + '\n'
    sock.sendall(login.encode('ASCII'))


def linesplit(socket):
    buffer = socket.recv(4096)
    buffering = True
    while buffering:
        if b'\n' in buffer:
            (line, buffer) = buffer.split(b'\n', 1)
            yield line + ""
        else:
            more = socket.recv(4096)
            if not more:
                buffering = False
            else:
                buffer += more
    if buffer:
        yield buffer


connect()
while True:
    data = linesplit(sock)

    if data:
        for line in data:
            print line
            # PROCESS MESSAGES HERE
    else:
        time.sleep(10)
        connect()
```

QUICK REFERENCE

Action	HTTP Request	Description
`List`	`GET https://ais.spire.com/vessels`	List all vessels.
`Get`	`GET https://ais.spire.com/vessels/{id}`	Get details for specific ship.
`List`	`GET https://ais.spire.com/messages`	List most recent AIS messages.

VESSELS API

Vessel Particulars

Vessel particulars include the highest-level information about a ship in our vessels database. The information contained here is a combination of data from AIS messages and external data sources. Each vessel goes through several layers of verification before making it to this list in an effort to provide a complete and up to date snapshot of the ship.

Property	Description
`id` *string*	Unique identifier of the vessel in the Spire database.
`name` *string*	Vessel name.
`mmsi` *integer*	Vessel Maritime Mobile Service Identity. Possible values: 000000000 to 999999999
`imo` *integer*	Vessel unique International Maritime Organization number. Possible values: 0001000000 to 0009999999, 0010000000 to 1073741823 (office flag state number)
`call_sign` *string*	Vessel call sign.

class *string*	Shipborne AIS transponder class. Valid values: A or B.
flag *string*	Vessel country flag using 2-letter country codes.
length *number*	Vessel length extracted from ship dimensions to_bow and to_stern in meters.
width *number*	Vessel width extracted from ship dimensions to_starboard and to_port in meters.
ais_version *integer*	Vessel AIS version. Valid values: 0 (compliant with Recommendation ITU-R M.1371-1), 1 (compliant with Recommendation ITU-R M.1371-3), 2 (compliant with Recommendation ITU-R M.1371-5 or later), 3 (compliant with future editions)
updated_at *date*	ISO8601 formatted date and time in UTC of the last time any field in the vessel record was updated.
general_classification *string*	Broad category of vessel.
individual_classification *string*	Specific category and cargo type of vessel.
person_capacity *integer*	Capacity of persons on board (passengers and crew).
gross_tonnage	A common measurement of the internal volume of a ship with

	which is equivalent to 100 cubic feet.
lifeboats *integer*	Indicates the number of lifeboats on board fitted with radio apparatus.

Last Known Position

The `last_known_position` object contains the positional information from the most recent AIS position message for a particular vessel.

Property	Description
maneuver *integer*	Vessel maneuver code. Valid values: 0 (not available; default), 1 (not engaged in special maneuver, 2 (engaged in special maneuver).
course *number*	Vessel course over ground.
draught *number*	Vessel draught represented in 1/10 meters. Possible values: 0.1 - 25.5, 0 (not available;default).
timestamp *date*	ISO8601 formatted AIS message timestamp in UTC.
rot *integer*	Vessel rate of turn. Possible values: -127 to 127; -128
geometry *object*	Vessel position coordinates represented in GeoJSON.
collection_type *string*	How the message was captured. Valid values: satellite or terrestrial.
speed	

heading *integer*	Vessel true heading represented in degrees.
accuracy *number*	Vessel GPS geolocation accuracy. Possible values: 1 (high, <10 m), 0 (low, >10m, default)

Predicted Position

The `predicted_position` object contains the predicted current positional information for a particular vessel.

Every 15 minutes the Predict engine runs and computes the predicted current location for most vessels in our database. Due to the nature of our data, there are a few instances in which we either do not calculate a predicted position at all or simply throw out our prediction:

- If we haven't seen a ship in over 48 hours.
- If we have received a new position report from the ship for a timestamp occurring after the time of our last prediction.
- If the point that we predicted is on land.

Property	Description
timestamp *date*	ISO8601 formatted time of position calculation timestamp in UTC.
geometry *object*	Calculated predicted vessel position coordinates represented in GeoJSON.
speed *number*	Predicted vessel speed over ground at time of calculation represented in knots.
course *number*	Predicted vessel course over ground at time of calculation.
confidence_radius	Estimated prediction error radius represented in nautical miles.

Most Recent Voyage

The `most_recent_voyage` object contains the voyage information for a particular vessel based on the most recent AIS information as entered by a ship's captain.

Property	Description
destination *string*	Destination as entered by vessel captain.
eta *date*	Estimated time of arrival date/time as entereed by vessel captain.

Filter Parameters

Property	Description	Supported Filters
mmsi *integer*	Vessel Maritime Mobile Service Identity. Possible values: 000000000 to 999999999.	Exact Match, List
imo *integer*	Vessel unique International Maritime Organization number. Possible values: 0001000000 to 0009999999, 0010000000 to 1073741823 (office flag state number).	Exact Match, List
call_sign *string*	Vessel call sign.	Exact Match, List
ship_type *string*	Category of vessel. Valid values: Fishing, Tug, Sailing, Pleasure Craft, Passenger, Cargo, Tanker, Other.	Exact Match, List
class *string*	Shipborne AIS transponder class. Valid values: A or B.	Exact Match, List
flag	Vessel country flag using 2-letter country codes.	Exac

updated_after *date*	Returns all vessels with an updated_at time greater or equal than the time specified.	Range
updated_before *date*	Returns all vessels with an updated_at time greater or equal than the time specified.	Range
arriving_after *date*	Returns all vessels with an eta time greater or equal than the time specified.	Range
arriving_before *date*	Returns all vessels with an eta time less or equal than the time specified.	Range
last_known_position_within *geometry*	Returns all vessels with a last_known_position point within provided GeoJSON polygon or multipolygon.	Geospatial
predicted_position_within *geometry*	Returns all vessels with a predicted_position point within provided GeoJSON polygon or multipolygon.	Geospatial
last_known_or_predicted_position_within *geometry*	Returns all vessels with a last_known_position OR predicted_position point within provided GeoJSON polygon or multipolygon.	Geospatial

MESSAGES API

Messages Resource

Field	Definition	Message Types
nmea *string*	Full NMEA 0183 v4 message.	All
msg_type		All

	Possible values: `1, 2, 3, 4, 5, 18, 19, 24, 27`	
`timestamp` *string*	ISO8601 formatted timestamp in UTC.	All
`created_at` *string*	ISO8601 formatted system ingestion time in UTC.	All
`mmsi` *integer*	Vessel MMSI. Possible values: 000000000 to 999999999	All
`collection_type` *string*	How the data was captured. Possible values: satellite or terrestrial	All
`msg_description` *string*	Description of the message type. Possible values: static, position, base_station, other	All
`source` *string*	Source satellite or base station, if available.	All
`msg_id` *string*	Unique identifier for each message. Created by combining the timestamp and MMSI.	All
`flag` *string*	Vessel country flag (derived from MMSI).	All
`flag_short_code` *string*	Vessel country flag short code (derived from MMSI).	All
`longitude` *number*	Vessel longitude in degrees (East = positive, West = negative).	1, 2, 3, 4,

`latitude` *number*	Vessel latitude in degrees (North = positive, South = negative). Possible values: +90 to -90; 91 (not available)	1, 2, 3, 4, 18, 19, 27
`position` *geometry*	Vessel position coordinates represented in GeoJSON.	1, 2, 3, 4, 18, 19, 27
`speed` *number*	Vessel speed over ground represented in knots. Possible values: 0 - 102.2 knots; 102.3 (not available)	1, 2, 3, 18, 19, 27
`course` *number*	Vessel course over ground in degrees. Possible values: 0 - 359.9; 360.0 (not available)	1, 2, 3, 18, 19, 27
`heading` *integer*	Vessel true heading represented in degrees. Possible values: 0 - 359; 511 (not available)	1, 2, 3, 18, 19
`status` *integer*	Vessel navigation status. Some common values: 0 (under way using engine), 1 (at anchor), 3 (restricted maneuverability), 7 (engaged in fishing), 15 (undefined, default)	1, 2, 3, 27
`accuracy` *integer*	Vessel GPS geolocation accuracy in meters. Possible values: 1 (high, <=10 m), 0 (low, >10m, default)	1, 2, 3, 4, 18, 19, 27
`rot` *integer*	Vessel rate of turn in degrees per minute. Possible values: -127 to 127; -128	1, 2, 3
`maneuver`	Vessel maneuver code.	1, 2,

ais_version *integer*	Vessel AIS version. Valid values: 0 (compliant with Recommendation ITU-R M.1371-1), 1 (compliant with Recommendation ITU-R M.1371-3), 2 (compliant with Recommendation ITU-R M.1371-5 or later), 3 (compliant with future editions)	5
name *string*	Vessel name.	5, 19, 24A
length *number*	Vessel width extracted from ship dimensions to_bow and to_stern in meters.	5, 19, 24B
width *number*	Vessel width extracted from ship dimensions to_starboard and to_port in meters.	5, 19, 24B
ship_and_cargo_type *integer*	Vessel ship and cargo type code. Some common values: 30 (fishing vessel), 52 (tug boat), 70 (cargo/fishing ship)	5, 19, 24B
ship_type *string*	Vessel ship and cargo type description.	5, 19, 24B
call_sign *string*	Vessel call sign.	5, 24B
imo *integer*	Vessel unique International Maritime Organization number. Possible values: 0 (not available; default), 0001000000-0009999999, 0010000000-1073741823 (office flag state number)	5

string		
eta *string*	Vessel estimated time of arrival represented as MMDDHHMM UTC. Possible values: Month: 1-12, 0 (not available; default); Day: 1-31, 0 (not available; default); Hour: 0-23, 24 (not available; default); Minute: 0-59, 60 (not available; default)	5
draught *number*	Vessel draught represented in 1/10 meters. Possible values: 0.1 - 255, 0 (not available;default)	5

Decoded Fields

By default, the Messages API returns a basic message format consisting of only the timestamp, NMEA message, & message ID. In order to receive all of the decoded fields and additional metadata, simply add it as a field filter parameter.

```
https://ais.spire.com/messages?fields=decoded
```

Filter Parameters

The Messages API is like a firehose for AIS messages. A basic query will return messages of all types, from all over the planet. This isn't entirely helpful for everyone as most people only care about messages from particular vessels, areas of interest, or sources.

To make narrowing this stream easier, we have introduced a variety of filters that helps you receive only the data you need.

Parameter	Description	Supported Filters
cleansed *bool*	Returns cleansed or uncleansed data. Valid values: true (default) or false	Exact Match

	Valid values: satellite or terrestrial	
msg_description *string*	Description of the message type. Valid values: static, position, base_station, other	Exact Match, List
msg_type *integer*	AIS message type. Possible values: 1, 2, 3, 4, 5, 18, 19, 24, 27	Exact Match, List
mmsi *integer*	Vessel MMSI. Possible values: 000000000 to 999999999	Exact Match, List
position *geometry*	Vessel position coordinates represented in GeoJSON.	Geospatial
received_after *date*	Returns valid AIS messages from a seven-day window with a timestamp greater than or equal to the time specified.	Range
received_before *date*	Returns valid AIS messages from a seven-day window with a timestamp less than or equal to the time specified.	Range

TCP FEED

Initiate Connection

To start receiving the feed, we must first whitelist your IP address(es), then you can connect to the provided Spire TCP feed server address on the specified port. Upon connecting, the server will wait to receive an authentication key. Pass the JWT in the following format in a single line.

A|T|<JWT Token>

The server will respond with error codes akin to the HTTP API if authentication fails or other issues arise when attempting to receive messages from the Spire TCP feed.

Note: depending on the system and program used, appending a new line " " might be required to make sure that the token is sent in one piece directly.

Feed Format

Data will be transferred line by line in NMEA v4 format. By default, timestamp tags will be added as shown below. Multiple line messages will be grouped and sent in order.

```
\c:1460667583*0a\!AIVDM,1,1,,A,13fA1P0020UdgRAti9ma47GB08Hf,0*2d
\c:1460667587*0e\!AIVDM,1,1,,A,16j?4r9P005>epWwWc6;uOwL0@K2,0*54
\c:1460667589*00\!AIVDM,1,1,,B,18156oP01k6SS9Mr@pEUS4GL08LK,0*38
\c:1460667591*09\!AIVDM,1,1,,B,16?UiaOP00UM;sN5dViP0?wT26Sd,0*60
\c:1460667601*03\!AIVDM,1,1,,B,19NSJ1P01P5g@9atiuB1p1Mn06`0,0*2f
\c:1460667608*0a\!AIVDM,1,1,,B,18156oP01k6SSGmr@n4UPTF@06`0,0*3e
\c:1460667614*07\!AIVDM,1,1,,A,177>?10022Ud8WuuJRR25AdN088g,0*08
\c:1460667620*00\!AIVDM,1,1,,A,15BA:p001PUkdHquTtp1wQ``00S0,0*38
```

Alternatively, customers can contact SPIRE's technical team to receive a feed without timestamps.

Reconnection Strategy

The server should try to keep the connection alive, up to 60 minutes. However, because satellite data is infrequent, time-outs might be reached on our server or on the client side.
In the event of a disconnection, the client can simply reconnect and authentify again using their secret token.

SUPPORT

SPIRE ANSWERS

Feeling stuck? Search our robust knowledge base for answers.

SYSTEM STATUS

Operational and uptime status of all Spire sytems. Subscribe to updates about scheduled maintanence and unplanned outages.

status.spire.com

CHAT & EMAIL

If you have any questions, issues, or feature suggestions don't hesitate to reach out to our Customer Experience team using the green chat icon in the bottom right-hand corner or send us an email at cx@spire.com.

PRIVACY POLICY