Accessing historical data via the IXON API v2 involves a series of steps to authenticate, discover endpoints, and then make the specific requests for the data you need.[1] Here's a breakdown of the process based on IXON's documentation:

Prerequisites:

1. **IXON Cloud Account and Licenses:** You need an IXON Cloud account with the "API / Integrations," "Data Studio," and "Live Data" modules enabled. If you don't have them, contact your IXON account manager or distributor.
2. **API Application ID:** You need to request an Application ID from IXON support.[2] This is crucial for all API v2 requests.
3. **Authentication Token (Bearer Token):** To make authenticated requests, you'll need to obtain a bearer token.[3] This typically involves sending your email and password (and possibly an OTP if 2FA is enabled) to the AccessTokenList endpoint.
4. **Historical Data Configuration:** Ensure that historical data logging is set up for your devices in the IXON Cloud. This means the IXrouter or IXagent is configured to send PLC data to the cloud, and variables are added as data tags.[4]

General Steps to Access Historical Data:

1. **Get Discovery (API Endpoints):**
   - The first step is to make a GET request to the base IXON API URL (e.g., https://portal.ixon.cloud:443/api/).
   - This "discovery" endpoint will return a JSON object containing a list of available endpoints and their corresponding URLs.[5] You'll use this to find the specific endpoints needed for historical data.
   - **Headers:** Include Api-Version: 2 and Api-Application: YOUR_APPLICATION_ID.
2. **Authenticate and Get Bearer Token:**
   - Use the AccessTokenList endpoint (obtained from discovery) to get your bearer token.
   - This is typically a POST request with a basic authorization header (your email and password base64 encoded, and potentially an OTP).[6]
   - The response will contain your bearer token, which you'll use in subsequent authenticated requests.
3. **Identify Your Company ID:**
   - Make a GET request to the CompanyList endpoint (from discovery).
   - This will give you the publicId of your company, which is often required in subsequent requests.
4. **Identify Your Agent (Device) ID:**
   - Make a GET request to the AgentList endpoint (from discovery).
   - Filter this list to find the specific agentId (which could be an IXrouter or IXagent) for which you want to retrieve historical data.
5. **Identify Data Sources and Tags:**
   - Once you have the agentId, you can query for data sources and tags associated with that agent.[7]
   - Use the AgentDataSourceList endpoint (from discovery) to get a list of data sources (e.g., PLCs) connected to your agent.[8] You'll need the publicId of the data source.
   - Use the AgentDeviceDataTagList endpoint (from discovery) to get a list of data tags (variables) that are being logged for historical data.[9] You'll need the tag ID (Int32) or slug (string) for the tags you want to retrieve.[10]
6. **Request Historical Data (DataList Endpoint):**
   - This is the core endpoint for retrieving historical data. It's typically a POST request to the DataList endpoint (from discovery).
   - **Request Body (JSON):** The request body is a JSON array containing one or more objects, each

specifying the data you want. Key parameters include:

- start: The start timestamp (ISO 8601 format, e.g., "2023-01-01T00:00:00Z").
- end: The end timestamp (ISO 8601 format).
- timeZone: The timezone for the timestamps (e.g., "UTC", or a tz database format like "America/New_York").
- source: An object containing the publicId of your data source.
- tags: An array of tag objects. For each tag:
  - slug or id: The identifier for the data tag.[11]
  - preAggr: Set to "raw" if you want the raw, unaggregated data.[12]
  - queries: An array of query objects for each tag, specifying:
    - ref: A reference for the query (e.g., the tag slug).
    - limit: The maximum number of data points to return (max 5000).[13]
    - offset: For pagination, to get subsequent sets of data points.
    - postAggr (optional): For post-aggregation (e.g., "mean", "median", "sum") if you want aggregated data.
    - step (optional): The number of seconds between subsequent data points when using post-aggregation.

- **Headers:** Include Api-Version: 2, Api-Application: YOUR_APPLICATION_ID, Authorization: Bearer YOUR_BEARER_TOKEN, and Api-Company: YOUR_COMPANY_ID.

7. **Process the Response:**
   - The DataList endpoint will return historical data in JSON format.[14] It will contain all requested values for a certain timestamp.
   - **Pagination:** The DataList endpoint's moreAfter attribute will always be null.[15] To get more data points beyond the limit, you need to use the offset parameter in your subsequent requests.

Example (Conceptual Python Request using requests library):

Python

```
import requests
import datetime

API_BASE_URL = 'https://portal.ixon.cloud:443/api/'
API_VERSION = '2'
YOUR_APPLICATION_ID = 'REPLACE_WITH_YOUR_APPLICATION_ID'
YOUR_COMPANY_ID = 'REPLACE_WITH_YOUR_COMPANY_ID'
YOUR_BEARER_TOKEN = 'REPLACE_WITH_YOUR_BEARER_TOKEN' # Obtain this first!

# 1. Get Discovery
headers = {
    'Api-Version': API_VERSION,
    'Api-Application': YOUR_APPLICATION_ID,
}
response = requests.get(API_BASE_URL, headers=headers)
response.raise_for_status() # Raise an exception for HTTP errors
```

```
discovery = {row['rel']: row['href'] for row in response.json()['data']}

# 2. (Assume you already have COMPANY_ID, AGENT_ID, DATA_SOURCE_ID, and TAG_SLUGS)
# If not, you'd make requests to CompanyList, AgentList, AgentDataSourceList, AgentDeviceDataTagList

# Example: Replace with your actual IDs/slugs
agent_id = 'YOUR_AGENT_PUBLIC_ID'
data_source_id = 'YOUR_DATA_SOURCE_PUBLIC_ID'
tag_slugs = ['your_tag_slug_1', 'your_tag_slug_2'] # Or use tag IDs

# Define the time range for historical data
end_time = datetime.datetime.utcnow().replace(microsecond=0)
start_time = end_time - datetime.timedelta(days=7) # Last 7 days

# 3. Request Historical Data
data_request_body = [
    {
        'start': start_time.isoformat() + 'Z',
        'end': end_time.isoformat() + 'Z',
        'timeZone': 'UTC', # Or your desired timezone, e.g., 'America/New_York'
        'source': {'publicId': data_source_id},
        'tags': [
            {
                'slug': tag_slug,
                'preAggr': 'raw', # Or 'mean', 'max', 'min', 'sum' if pre-aggregated
                'queries': [
                    {
                        'ref': tag_slug,
                        'limit': 5000, # Max limit per request
                        'offset': 0
                        # 'postAggr': 'mean', # Optional: for post-aggregation
                        # 'step': 3600 # Optional: for step when post-aggregating (e.g., hourly mean)
                    }
                ]
            }
            for tag_slug in tag_slugs
        ]
    }
]

historical_data_headers = {
    'Api-Version': API_VERSION,
    'Api-Application': YOUR_APPLICATION_ID,
    'Authorization': f'Bearer {YOUR_BEARER_TOKEN}',
    'Api-Company': YOUR_COMPANY_ID,
    'Content-Type': 'application/json'
}

response = requests.post(
```

```
    discovery['DataList'], # Use the DataList URL from discovery
    headers=historical_data_headers,
    json=data_request_body
)
response.raise_for_status()

historical_data = response.json()
print(historical_data)

# To handle pagination (if you need more than 5000 points per tag/query):
# You would repeat the DataList request, incrementing the 'offset'
# by the 'limit' (e.g., 5000, 10000, etc.) until no more data is returned.
```

Important Notes:

- **Error Handling:** Always implement robust error handling (e.g., checking response.status_code and handling exceptions).
- **Rate Limiting:** Be aware of IXON's API rate limits. Implement a handler for HTTP 429 "Too Many Requests" responses by dynamically reducing the number of requests or implementing retries with exponential backoff.
- **Timezones:** Pay close attention to timezones (timeZone parameter) to ensure you're requesting and interpreting data correctly. IXON typically uses UTC by default if no timezone is specified.
- **Documentation:** Always refer to the official IXON Developer Documentation for the most up-to-date and detailed information on API endpoints, request parameters, and response formats. The structure of the DataList request and the parameters within tags and queries are critical.
- **Export to CSV:** If you need to export large amounts of raw data to a CSV file, IXON recommends using the DataExport endpoint instead of DataList.[16] This endpoint is designed for bulk export.

By following these steps, you can effectively use the IXON API v2 to programmatically access and retrieve your historical machine data for integration into your own applications or dashboards.