You're looking to interact with IXON's IXrouter or IXagent devices using their API v2. The IXON API allows you to retrieve information about your devices (agents), manage them, and access their data (live or historical).

The core concept is to use the **AgentList** endpoint to identify your specific IXrouter or IXagent, and then potentially other endpoints for more detailed device management or data retrieval.

Here's a breakdown and a Python example:

Prerequisites:

1. **IXON Cloud Account and API/Integrations Module:** Ensure you have the necessary IXON Cloud account and the "API / Integrations" module enabled.
2. **API Application ID:** You need an Api-Application ID. Contact IXON support to obtain this.
3. **Authentication (Bearer Token):** You'll need to obtain a Bearer Token by authenticating with your IXON Cloud credentials. This token is used in the Authorization header for most API requests.
4. **Company ID:** You'll need your Api-Company ID. This can be obtained via the CompanyList endpoint.

Steps to Access IXrouter/IXagent Information:

1. **Discovery:** As with all IXON API v2 interactions, start by making a GET request to the base API URL (https://portal.ixon.cloud:443/api/) to get the current list of available endpoints. This is crucial as endpoint URLs can change.
2. **Authentication:** Obtain your bearer token.
3. **Get Company ID:** If you don't already know it, get your company's publicId from the CompanyList endpoint.
4. **List Agents (IXrouters/IXagents):** Use the AgentList endpoint (from your discovery result) to retrieve a list of all IXrouters and IXagents associated with your company.
5. **Filter/Inspect Agent Details:** Once you have the list of agents, you can inspect their details (name, online status, publicId, etc.) or filter them to find a specific device. You can also request specific fields to limit the amount of data returned.
6. **Perform Actions (Optional):** Once you have an agentId, you can use other endpoints (e.g., Agent with PATCH for configuration, or DataList for historical data as discussed previously) to interact with that specific device.

Example: Listing your IXrouters/IXagents in Python

This example demonstrates how to:

1. Perform API discovery.
2. Obtain a bearer token (conceptual, as this involves your credentials).
3. Get your company ID.
4. List all agents (IXrouters/IXagents) under your company.

Python

```python
import requests
import json

# --- Configuration (REPLACE THESE WITH YOUR ACTUAL VALUES) ---
API_BASE_URL = 'https://portal.ixon.cloud:443/api/'
API_VERSION = '2'
YOUR_APPLICATION_ID = 'YOUR_IXON_API_APPLICATION_ID' # Get this from IXON support

# You would typically obtain this dynamically after user login or from a secure store.
# For demonstration purposes, replace with a valid token.
YOUR_BEARER_TOKEN = 'YOUR_OBTAINED_BEARER_TOKEN'

# You would typically obtain this dynamically from the CompanyList endpoint
# For demonstration purposes, replace with your company's publicId.
YOUR_COMPANY_ID = 'YOUR_IXON_COMPANY_PUBLIC_ID'

# --- Helper Function for API Requests ---
def make_ixon_api_request(method, url, headers, json_data=None):
    try:
        response = requests.request(method, url, headers=headers, json=json_data)
        response.raise_for_status()  # Raise an HTTPError for bad responses (4xx or 5xx)
        return response.json()
    except requests.exceptions.HTTPError as e:
        print(f"HTTP Error: {e.response.status_code} - {e.response.text}")
        return None
    except requests.exceptions.RequestException as e:
        print(f"Request Error: {e}")
        return None

# --- Main Script ---
def access_ixon_agents():
    # 1. API Discovery
    print("1. Performing API Discovery...")
    discovery_headers = {
        'Api-Version': API_VERSION,
        'Api-Application': YOUR_APPLICATION_ID,
    }
    discovery_response = make_ixon_api_request('GET', API_BASE_URL, discovery_headers)

    if not discovery_response:
        print("Failed to perform API discovery. Exiting.")
        return

    discovery_endpoints = {row['rel']: row['href'] for row in discovery_response['data']}
```

```
    print("Discovery successful. Available endpoints:")
    for rel, href in discovery_endpoints.items():
        print(f"  - {rel}: {href}")

    # 2. (Conceptual) Obtain Bearer Token and Company ID
    # In a real application, you'd use the AccessTokenList endpoint to get the token,
    # and CompanyList endpoint to get the Company ID.
    # For this example, we assume YOUR_BEARER_TOKEN and YOUR_COMPANY_ID are already set.
    print("\n2. (Assuming Bearer Token and Company ID are already set from previous steps)")
    if not YOUR_BEARER_TOKEN or not YOUR_COMPANY_ID:
        print("Please ensure YOUR_BEARER_TOKEN and YOUR_COMPANY_ID are configured.")
        print("You'll need to implement token acquisition via AccessTokenList and company ID via CompanyList.")
        return

    # 3. List Agents (IXrouters/IXagents)
    print("\n3. Retrieving list of IXrouters/IXagents...")
    agent_list_url = discovery_endpoints.get('AgentList')
    if not agent_list_url:
        print("AgentList endpoint not found in discovery. Cannot proceed.")
        return

    agent_list_headers = {
        'Api-Version': API_VERSION,
        'Api-Application': YOUR_APPLICATION_ID,
        'Authorization': f'Bearer {YOUR_BEARER_TOKEN}',
        'Api-Company': YOUR_COMPANY_ID,
    }

    # You can add filters and fields to the URL if needed.
    # Example: filter by online status and get specific fields
    # agent_list_url += '?filters=eq(connectedUserCount,1)&fields=name,publicId,online'

    agents_response = make_ixon_api_request('GET', agent_list_url, agent_list_headers)

    if agents_response and agents_response['data']:
        print("\nIXON Agents (IXrouters/IXagents) found:")
        for agent in agents_response['data']:
            print(f"  - Name: {agent.get('name', 'N/A')}")
            print(f"    Public ID: {agent.get('publicId', 'N/A')}")
            print(f"    Online: {agent.get('online', 'N/A')}") # 'online' field often indicates connection status
            # You can access other fields returned by the API here
            print("-" * 30)
    elif agents_response:
        print("No IXON Agents found for your company.")
    else:
```

```
        print("Failed to retrieve agent list.")

if __name__ == "__main__":
    access_ixon_agents()
```

How to get YOUR_OBTAINED_BEARER_TOKEN and YOUR_IXON_COMPANY_PUBLIC_ID:

In a real scenario, you'd execute these steps before the agent listing:

1. Getting your Bearer Token:

Python

```
# ... (imports and API_BASE_URL, API_VERSION, YOUR_APPLICATION_ID defined)

# This part would typically be for an initial login or token refresh
def get_bearer_token(email, password, application_id, discovery_endpoints):
    access_token_url = discovery_endpoints.get('AccessTokenList')
    if not access_token_url:
        print("AccessTokenList endpoint not found.")
        return None

    auth_headers = {
        'Api-Version': API_VERSION,
        'Api-Application': application_id,
        'Content-Type': 'application/json'
    }
    auth_payload = {
        'email': email,
        'password': password
        # If 2FA is enabled, you might need 'otp': 'your_otp_code'
    }

    print("Attempting to get bearer token...")
    response = requests.post(access_token_url, headers=auth_headers, json=auth_payload)
    if response.status_code == 200:
        token_data = response.json()
        print("Bearer token obtained successfully.")
        return token_data['data']['secretId'] # Or 'data.secretId' depending on exact structure
    else:
        print(f"Failed to get bearer token: {response.status_code} - {response.text}")
        return None
```

```
# ... in your main function, after discovery:
# my_email = "your_email@example.com"
# my_password = "your_password"
# YOUR_BEARER_TOKEN = get_bearer_token(my_email, my_password, YOUR_APPLICATION_ID, discovery_endpoints)
# if not YOUR_BEARER_TOKEN:
#     return # Exit if token acquisition failed
```

2. Getting your Company ID:

Python

```
# ... (imports, API_BASE_URL, API_VERSION, YOUR_APPLICATION_ID, YOUR_BEARER_TOKEN defined)

def get_company_id(application_id, bearer_token, discovery_endpoints):
    company_list_url = discovery_endpoints.get('CompanyList')
    if not company_list_url:
        print("CompanyList endpoint not found.")
        return None

    company_headers = {
        'Api-Version': API_VERSION,
        'Api-Application': application_id,
        'Authorization': f'Bearer {bearer_token}',
    }

    print("Retrieving company list...")
    response = requests.get(company_list_url, headers=company_headers)
    if response.status_code == 200:
        companies_data = response.json()
        if companies_data['data']:
            # Assuming you want the first company if you only have one.
            # You might need logic to select the correct company if you have multiple.
            company_id = companies_data['data'][0]['publicId']
            print(f"Company ID obtained: {company_id}")
            return company_id
        else:
            print("No companies found.")
            return None
    else:
```

```
        print(f"Failed to get company list: {response.status_code} - {response.text}")
        return None

# ... in your main function, after token acquisition:
# YOUR_COMPANY_ID = get_company_id(YOUR_APPLICATION_ID, YOUR_BEARER_TOKEN,
discovery_endpoints)
# if not YOUR_COMPANY_ID:
#     return # Exit if company ID acquisition failed
```

What you can do with IXrouter/IXagent data:

Once you have the publicId of an IXrouter or IXagent, you can:

- **Get Detailed Agent Information:**
  ```
  Python
  # Example: Get specific fields for a single agent
  single_agent_url = f"{discovery_endpoints['Agent']}/{agent_public_id}?
  fields=name,online,networkValues,connectedUsers"
  agent_details = make_ixon_api_request('GET', single_agent_url, agent_list_headers)
  print(f"\nDetails for {agent_public_id}:")
  print(json.dumps(agent_details, indent=2))
  ```
  (Note: The Agent endpoint might be https://portal.ixon.cloud:443/api/agents/{agentId} as per some IXON examples).
- **Access Live Data:** Use the DataLive endpoint to get real-time values from the device's configured data tags.
- **Access Historical Data:** Use the DataList endpoint as explained in the previous answer to retrieve logged data.
- **Manage Device Configuration:** Use PATCH or PUT requests to the Agent endpoint (and potentially other related configuration endpoints) to change device settings.
- **Manage VPN Connections:** Use the VPN Client API to programmatically connect to an IXrouter/IXagent.

Remember to consult the official IXON Developer Documentation for the most accurate and up-to-date information on specific endpoints and their required parameters.

Sources

1. https://github.com/jackskii/gpt
2. https://github.com/ratulalahy/uvu_advisor_bot_client
3. https://github.com/Fundacion-Universidad-de-Antioquia/conexionerp