**Catastrophe Parameter Extraction Microservice – Technical Overview**

**1. Purpose**

**This microservice extracts forecast parameters for catastrophic events based on geographic coordinates and dates, producing structured JSON output. It integrates tightly with the Catastrophe Detection Microservice, providing automatic parameter updates (pressure, temperature, humidity, wind speed, etc.) for detected events, and ensures that the catastrophe database is updated reliably and efficiently.**

**2. Scope**

* **Input: geographic coordinates (single or multiple) and a date.**
* **Output: structured forecast parameters per coordinate, including nearest grid point matching.**
* **Automation: invoked automatically by the Catastrophe Detection Microservice for each newly detected event.**
* **Integration: Python-based processing service that reads NetCDF (.nc) datasets (e.g., GEOS-FP) and returns 10-day forecasts for relevant meteorological parameters.**

**3. Inputs**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| **coords** | **Array of Objects** | **Each object must include lat and long** |
| **date** | **String** | **Date in YYYY-MM-DD format** |

**Constraints & Notes:**

* **Latitude: [-90, 90], Longitude: [-180, 180]**
* **Multiple coordinate points allowed; each is processed independently.**
* **Dates must be valid ISO date strings.**

**4. Outputs**

**The service returns a JSON object, structured per parameter and coordinate:**

**{**

**"PS": [**

**{**

**"original\_coord": {"lat": 29.97, "long": 31.14},**

**"grid\_point": {"lat": 30.0, "lon": 31.25},**

**"forecast": {**

**"day\_0": {"hour\_offset": 0, "PS": 1013},**

**"day\_1": {"hour\_offset": 24, "PS": 1012},**

**...**

**}**

**}**

**],**

**"TS": [...],**

**"QV2M": [...],**

**"TQV": [...],**

**"U10M": [...],**

**"V10M": [...]**

**}**

**Notes:**

* **Forecast is generated for 10 days with 24-hour resolution.**
* **Each coordinate is mapped to the nearest grid point in the NetCDF dataset.**
* **Missing or invalid coordinates default to the nearest valid grid point.**

**5. Automation & Integration**

1. **Catastrophe Detection Microservice identifies new events.**
2. **Coordinates & date for each event are sent to this microservice.**
3. **Forecast parameters are extracted using NetCDF datasets.**
4. **Results are returned as JSON and automatically stored in the catastrophe database.**

**Data Flow:**

**[Catastrophe Detection] --> [Parameter Extraction Service] --> [Database Update]**

**6. Use Cases**

| **Use Case** | **Input** | **Expected Output** |
| --- | --- | --- |
| **Valid Input** | **{coords:[{lat:29.97,long:31.14}], date:'2025-10-04'}** | **10-day forecast for PS, TS, QV2M, TQV, U10M, V10M** |
| **Multiple Coords** | **Two or more coordinate points** | **Forecast for each coordinate included in output** |
| **Invalid Coords** | **{lat:999,long:999}** | **Nearest valid grid point is used** |
| **Invalid Date Format** | **date:'04-10-2025'** | **JSON error returned indicating invalid date format** |
| **Missing Parameter** | **File not found or parameter missing** | **JSON error returned indicating missing required data** |

**7. Error Handling**

* **Invalid Coordinates: Adjust to nearest valid grid point; log warning.**
* **Invalid Date: Return structured error JSON, reject processing.**
* **Missing Parameter / File: Return structured error JSON and log incident.**
* **Dataset Read Errors: Retry extraction; log any failures.**

**8. Non-Functional Requirements**

* **Performance: 10+ coordinate forecasts processed under 1 second per coordinate.**
* **Reliability: Ensure consistent extraction and database updates even under multiple concurrent requests.**
* **Scalability: Can process multiple events in parallel, supporting high-throughput catastrophe detection scenarios.**
* **Maintainability: Python module is modular, supporting future extensions (new parameters, datasets, resolutions).**

**9. Future Enhancements**

* **Integration with real-time satellite feeds for higher temporal resolution.**
* **Support additional parameters (e.g., rainfall intensity, soil moisture).**
* **Adaptive forecasting using ML models on historical data.**
* **Parallelization and distributed computation for large-scale multi-coordinate extractions.**