DVOACAP API - QUICK REFERENCE

JSON FORMAT (What the DLL expects)

✓ CORRECT FORMAT:

```
{
  "Arguments": {
    "RxLocations": [
      {"Lat": 44.90, "Lon": 20.50, "Label": "BELGRADE"}
    ],
    "Hours": {"Start": 1, "Step": 1, "Count": 24},
    "Freqs": [6.07, 7.20, 9.70, 11.85, 13.70, 15.35, 17.73, 21.65, 25.89],
    "IncludeMuf": true
  }
}
```

✗ OLD FORMAT (DOESN'T WORK):

```
{
  "Method": 30,
  "Transmitter": {"Latitude": 44.374, "Longitude": -64.300},
  "Receiver": {"Latitude": 44.90, "Longitude": 20.50},
  "Frequencies": [14.15],
  "Hours": [18]
}
```

PYTHON API METHODS

1. SIMPLE SINGLE PREDICTION

```
   engine.predict_simple(
       tx_lat, tx_lon,       # Your location
       rx_lat, rx_lon,       # Target location
       frequency_mhz,        # Single frequency
       hour_utc,             # Single hour (0-23)
       ssn,                  # Sunspot number
       rx_label="RX"         # Optional label
   )
   → Returns: {'reliability': int, 'snr_db': float, 'quality': str, 'muf': float}
```

2. MULTI-BAND 24-HOUR ANALYSIS

```
  engine.predict_multi_band(
      tx_lat, tx_lon,
      rx_lat, rx_lon,
      rx_label="LOCATION",
      ssn=140
  )
  → Predicts: 160m-10m for all 24 hours
  → Returns: Full DVOACAP result structure
```

3. FULL CONTROL (Advanced)

```
  engine.predict(
      tx_lat, tx_lon,
      rx_locations=[{"Lat": ..., "Lon": ..., "Label": "..."}],
      frequencies=[7.1, 14.15, 21.2],
      hours={"Start": 0, "Step": 1, "Count": 24},
      ssn=140,
      month=None,          # Optional (defaults to current)
      include_muf=True
  )
  → Returns: Full DVOACAP result with all requested data
```

EXAMPLE USAGE

```
from dvoacap_wrapper import DVOACAPEngine

# Initialize
engine = DVOACAPEngine("dvoa.dll")

# Quick prediction
result = engine.predict_simple(
    tx_lat=44.374, tx_lon=-64.300,  # Halifax
    rx_lat=51.5, rx_lon=-0.1,       # London
    frequency_mhz=14.15,            # 20m
    hour_utc=18,                    # 18:00 UTC
    ssn=140                         # High solar activity
)

print(f"{result['quality']}: {result['reliability']}% reliable, SNR={result['snr_db']}dB")
# Output: GOOD: 85% reliable, SNR=12.3dB
```

INTEGRATION WITH GENERATE SCRIPT

Run with DVOACAP:

```
    python generate_propagation_voacap.py --dvoacap
```

Run with ITU-R (fast):

```
    python generate_propagation_voacap.py
```

The script automatically:
- Uses DVOACAP if available and requested
- Falls back to ITU-R if DVOACAP fails
- Compares both methods (saves ITU-R as baseline)

KEY PARAMETER NOTES

RxLocations:
- Array of objects with Lat, Lon, Label
- Can include multiple locations in one call
- Label is used for identification in results

Hours:
- Start: Starting hour (0-23)
- Step: Hour increment (usually 1)
- Count: Number of hours to predict

Freqs:
- Array of frequencies in MHz
- Common bands: 1.85, 3.65, 7.1, 10.13, 14.15, 18.1, 21.2, 24.95, 28.4
- Can be any frequency 1-30 MHz

SSN (Sunspot Number):
- 0-300 typical range
- 100-150 = moderate activity
- 150-200 = high activity
- Check current: https://www.swpc.noaa.gov/

RETURN VALUES
----

predict_simple() returns:

```
{
    'reliability': 85,          # 0-100%
    'snr_db': 12.3,             # Signal-to-noise ratio
    'quality': 'GOOD',          # GOOD/FAIR/POOR/CLOSED
    'muf': 18.5,                # Maximum Usable Frequency
    'method': 'VOACAP'          # Prediction method used
}
```

Quality levels:
- GOOD: reliability >= 70%
- FAIR: reliability >= 40%
- POOR: reliability < 40%
- CLOSED: frequency > MUF

TESTING

Test the wrapper:

```
    python dvoacap_wrapper.py
```

Should output:

```
    ✓ DVOACAP engine loaded from dvoa.dll
    [TEST 1] Single path/frequency/hour
      Method: VOACAP
      Quality: GOOD
      Reliability: 85%
      SNR: 12.3 dB
      MUF: 18.5 MHz
    [TEST 2] Multi-band 24-hour prediction
      ✓ Received full prediction data
    ✓ Tests complete!
```

TROUBLESHOOTING

Import Error:
    → Ensure dvoacap_wrapper.py is in same directory
    → Check: from dvoacap_wrapper import DVOACAPEngine

DLL Not Found:

→ Download dvoa.dll from https://github.com/VE3NEA/DVOACAP

→ Place in same directory as script

Zero/Invalid Results:

→ Check SSN is reasonable (0-300)

→ Verify coordinates are correct

→ Try different frequency/hour combination

Parse Error:

→ DVOACAP result format may differ from expected

→ Check raw result in error message

→ Update parsing logic in predict_simple()