MoveAxis Satellite Tracking - Integration Guide

Overview

This guide shows you how to integrate the corrected MoveAxis implementation into your existing OnStepX Alpaca bridge for full satellite tracking support.

Changes Required

1. Update `telescope.py`

Location: `~/alpaca-onstepx/telescope.py`

A. Add to `__init__` method:

```python
def __init__(self, ...):
    # ... existing initialization ...

    # Add these lines:
    self._max_axis_rate = 2.0  # Default max rate, degrees/second
```

B. Replace the `move_axis()` method:

Find this (INCORRECT implementation):

```python
def move_axis(self, axis, rate):
    """Move axis at specified rate"""
    if axis == TelescopeAxes.axisPrimary:  # RA
        if rate > 0:
            self.send_command(f':RG{abs(rate):.1f}#')  # WRONG!
            self.send_command(':Me#')
        # ...
```

Replace with:

python

```python
def move_axis(self, axis, rate):
    """
    Move telescope axis at specified rate (ASCOM Platform 7 compliant)

    Supports variable-rate movement for satellite tracking.
    Both axes can be commanded simultaneously.

    Args:
        axis: TelescopeAxes.axisPrimary (RA) or TelescopeAxes.axisSecondary (Dec)
        rate: Rate in degrees per second
            Positive: East (RA) or North (Dec)
            Negative: West (RA) or South (Dec)
            Zero: Stop movement
    """
    if not self.is_connected:
        return

    if axis == TelescopeAxes.axisPrimary:
        self._move_ra_axis(rate)
    elif axis == TelescopeAxes.axisSecondary:
        self._move_dec_axis(rate)

def _move_ra_axis(self, rate):
    """Move RA axis at specified rate"""
    if rate == 0:
        self.send_command(':Qe#')
        self.send_command(':Qw#')
        return

    # Set variable rate
    abs_rate = abs(rate)
    self.send_command(f':RA{abs_rate:.4f}#')

    # Start movement
    if rate > 0:
        self.send_command(':Me#')  # East
    else:
        self.send_command(':Mw#')  # West

def _move_dec_axis(self, rate):
    """Move Dec axis at specified rate"""
    if rate == 0:
        self.send_command(':Qn#')
        self.send_command(':Qs#')
        return

    # Set variable rate
    abs_rate = abs(rate)
    self.send_command(f':RE{abs_rate:.4f}#')

    # Start movement
    if rate > 0:
        self.send_command(':Mn#')  # North
    else:
        self.send_command(':Ms#')  # South
```

C. Add new helper methods:

```python
def get_axis_rates(self, axis):
    """Get available rates for axis (ASCOM compliance)"""
    if not hasattr(self, '_max_axis_rate'):
        self._max_axis_rate = 2.0

    return [{
        'Minimum': 0.0,
        'Maximum': self._max_axis_rate
    }]

def set_satellite_tracking_rates(self, ra_rate, dec_rate):
    """Convenience: set both axis rates simultaneously"""
    self.move_axis(TelescopeAxes.axisPrimary, ra_rate)
    self.move_axis(TelescopeAxes.axisSecondary, dec_rate)

def stop_all_movement(self):
    """Emergency stop: halt all movement"""
    self.send_command(':Q#')
    self.send_command(':Qe#')
    self.send_command(':Qw#')
    self.send_command(':Qn#')
    self.send_command(':Qs#')
```

---

2. Update `main.py` (API Routes)

Location: `~/alpaca-onstepx/main.py`

Add/Update MoveAxis endpoint:

```python
@app.route('/api/v1/telescope/0/moveaxis', methods=['PUT'])
@helpers.require_connected('telescope')
def telescope_moveaxis():
    """Move telescope axis at specified rate"""
    axis = helpers.get_form_value('Axis', 0, int)
    rate = helpers.get_form_value('Rate', 0.0, float)

    # Validate axis
    if axis not in [0, 1]:  # 0=Primary/RA, 1=Secondary/Dec
        return helpers.alpaca_error(
            config.ASCOM_ERROR_CODES['INVALID_VALUE'],
            f"Invalid axis: {axis}"
        )

    # Move axis
    telescope.move_axis(axis, rate)
    return helpers.alpaca_response(None)

@app.route('/api/v1/telescope/0/axisrates')
@helpers.require_connected('telescope')
def telescope_axisrates():
    """Get available rates for specified axis"""
    axis = helpers.get_form_value('Axis', 0, int)

    rates = telescope.get_axis_rates(axis)
    return helpers.alpaca_response(rates)

@app.route('/api/v1/telescope/0/canmoveaxis', methods=['GET'])
def telescope_canmoveaxis():
    """Check if MoveAxis is supported"""
    axis = helpers.get_form_value('Axis', 0, int)

    # Both axes supported
    can_move = axis in [0, 1]
    return helpers.alpaca_response(can_move)
```

---

3. Update Capability Reporting

Location: `~/alpaca-onstepx/main.py`

Ensure these endpoints return correct values:

```python
@app.route('/api/v1/telescope/0/canmoveaxis', methods=['GET'])
def telescope_canmoveaxis():
    """MoveAxis capability"""
    axis = helpers.get_form_value('Axis', 0, int)
    return helpers.alpaca_response(axis in [0, 1])

@app.route('/api/v1/telescope/0/canpulseguide')
def telescope_canpulseguide():
    """Pulse guide capability"""
    return helpers.alpaca_response(True)
```

Testing

Quick Test (Manual)

```bash
# 1. Connect to your OnStepX
cd ~/alpaca-onstepx
source venv/bin/activate

# 2. Start Python interpreter
python3

# 3. Test commands
>>> import socket
>>> s = socket.socket()
>>> s.connect(('192.168.1.100', 9999))  # Your OnStepX IP

>>> # Test RA rate
>>> s.send(b':RA0.5#')
>>> s.recv(1024)

>>> # Test Dec rate
>>> s.send(b':RE0.3#')
>>> s.recv(1024)

>>> # Start movement
>>> s.send(b':Me#')
>>> s.send(b':Mn#')

>>> # Stop after a few seconds
>>> import time
>>> time.sleep(3)
>>> s.send(b':Qe#')
>>> s.send(b':Qn#')
>>> s.close()
```

Full Test Suite

```bash
# Run the comprehensive test
python test_moveaxis_satellite.py 192.168.1.100
```

Using with Satellite Tracking Software

Configuration

Most satellite tracking software expects:

1. **ASCOM Telescope driver** - Your Alpaca bridge provides this
2. **MoveAxis support** - Now correctly implemented
3. **Tracking Mode**: Set to "Continuous" or "MoveAxis"

Example Software

1. SkyTrack

```
Settings:
- Telescope: ASCOM Alpaca
- Server: http://raspberrypi.local:5555
- Tracking Method: Continuous (MoveAxis)
- Update Rate: 1 second
```

2. WinSatTrack

```
Settings:
- Mount Type: ASCOM
- Driver: Alpaca Telescope
- Update Method: MoveAxis
```

3. Custom Python Script

```python
from alpaca.telescope import Telescope
import satellite_predictor  # Your satellite library

# Connect
telescope = Telescope('raspberrypi.local:5555', 0)
telescope.Connected = True

# Track satellite
while tracking:
    # Get current satellite position/velocity
    ra, dec, ra_rate, dec_rate = get_satellite_state()

    # Command mount
    telescope.MoveAxis(0, ra_rate)   # RA axis
    telescope.MoveAxis(1, dec_rate)  # Dec axis

    time.sleep(1)  # Update every second

# Stop
telescope.MoveAxis(0, 0)
telescope.MoveAxis(1, 0)
```

Troubleshooting

Problem: Mount doesn't move

Check:

1. Tracking enabled? (`telescope.Tracking = True`)
2. Mount limits? (Near horizon or zenith?)
3. Rate too high? (Try 0.1°/s first)

Test:

```bash
# Minimal test
curl -X PUT "http://raspberrypi.local:5555/api/v1/telescope/0/moveaxis" \
  -d "Axis=0" \
  -d "Rate=0.1"
```

Problem: Movement jerky/stuttering

Causes:

- Network latency
- Update rate too fast
- Mount not designed for continuous variable rates

Fix:

- Increase update interval (2-3 seconds)
- Smooth rate changes
- Check OnStepX configuration

Problem: Mount won't track fast-moving satellites

Limits:

- OnStepX MaxRate is mount-dependent
- Typical: 1-4 degrees/second
- ISS at zenith can exceed this!

Solution:

- Use alt-az mount for overhead passes
- Avoid high-altitude passes for equatorial mounts
- Check mount specs

Command Reference

OnStepX Commands Used

Command	Purpose	Parameters	Response
`:RAn.n#`	Set RA rate	n.n = deg/sec	0 or 1
`:REn.n#`	Set Dec rate	n.n = deg/sec	0 or 1
`:Me#`	Move East	None	None
`:Mw#`	Move West	None	None
`:Mn#`	Move North	None	None
`:Ms#`	Move South	None	None
`:Qe#`	Stop East	None	None
`:Qw#`	Stop West	None	None
`:Qn#`	Stop North	None	None
`:Qs#`	Stop South	None	None
`:Q#`	Emergency Stop	None	None

ASCOM API Endpoints

Endpoint	Method	Parameters	Returns
`/api/v1/telescope/0/moveaxis`	PUT	Axis, Rate	None
`/api/v1/telescope/0/axisrates`	GET	Axis	Rate array
`/api/v1/telescope/0/canmoveaxis`	GET	Axis	Boolean

Pulse Guiding Confirmation ✓

Your pulse guiding implementation **is correct**:

```python
def pulse_guide(self, direction, duration_ms):
    direction_cmds = {
        GuideDirections.guideNorth: ':Mgn',
        GuideDirections.guideSouth: ':Mgs',
        GuideDirections.guideEast: ':Mge',
        GuideDirections.guideWest: ':Mgw',
    }

    cmd = f"{direction_cmds[direction]}{duration_ms:04d}#"
    self.send_command(cmd)
```

This correctly uses the OnStepX `:Mgdnnnn#` format where:

- `d` = direction (n/s/e/w)
- `nnnn` = duration in milliseconds (20-16399ms)

No changes needed for pulse guiding! ✓

Summary

What Changed

✅ MoveAxis now uses variable rate commands (`:RAn.n#`, `:REn.n#`)
✅ Both axes can move simultaneously
✅ Rates are in degrees/second as required by ASCOM
✅ Full satellite tracking support

What Stayed the Same

✅ Pulse guiding implementation (already correct)
✅ All other telescope functionality
✅ API endpoints (just corrected behavior)

Ready For

✅ Satellite tracking software
✅ Custom tracking rates
✅ Advanced mount control
✅ Simultaneous multi-axis movement

Your OnStepX Alpaca bridge is now ready for professional satellite tracking! 🛰️