Dell Unisphere Client Library

A Robust Interface for Unity Storage Management

Purpose and Overview

- Provides a standardized programmatic interface to Dell Unisphere REST API
- Handles authentication, session management, and API interaction
- Supports complete software upgrade workflow
- Simplifies interaction with Dell Unity storage systems
- Offers both programmatic and CLI access

Client Architecture


Client Architecture

- **Core Client**: Main interface for all operations
- **Session Management**: Handles authentication and session persistence
- **API Layer**: Specialized clients for different API domains
- **Error Handling**: Comprehensive exception handling
- **CLI Interface**: Command-line access to all functionality

Key Components

- **UnisphereClient**: Main entry point for all operations
- **SessionManager**: Handles session creation, storage, and validation
- **API Modules**:
  - SystemApi: System information and configuration
  - SoftwareApi: Software version management
  - UpgradeApi: Software upgrade operations
- **Exception Classes**: Specialized error types

Authentication Flow

```
# Create a client
client = UnisphereClient(
    base_url="http://unisphere.example.com",
    username="admin",
    password="Password123!",
    verify_ssl=True
)

# Login happens automatically with first API call
# or can be done explicitly
client.login()

# Client maintains session across API calls

# Logout when done
client.logout()
```

Session Management

- Automatic session creation on first API call
- CSRF token handling for secure API interaction
- Session persistence to avoid repeated authentication
- Automatic session renewal when expired
- Context manager support (`with` statement)

Basic System Information

```
# Create client instance
with UnisphereClient("https://unisphere.example.com", "admin", "Password123!") as client:
    # Get system information
    system_info = client.get_basic_system_info()

    # Extract system details
    system_model = system_info["entries"][0]["content"]["model"]
    system_name = system_info["entries"][0]["content"]["name"]
    software_version = system_info["entries"][0]["content"]["softwareVersion"]

    print(f"System: {system_name} ({system_model})")
    print(f"Software Version: {software_version}")
```

Software Management

```
# Get installed software version
installed_version = client.get_installed_software_version()

# Upload a software package
upload_result = client.upload_package("/path/to/Unity_5.4.0.bin")
file_id = upload_result["id"]

# Prepare software from uploaded package
prepare_result = client.prepare_software(file_id)
candidate_id = prepare_result["id"]

# Get all candidate software versions
candidates = client.get_candidate_software_versions()
```

Upgrade Workflow

1. Upload software package
2. Prepare candidate software version
3. Verify upgrade eligibility
4. Create upgrade session
5. Monitor upgrade progress
6. Handle pauses/resumptions (if needed)

Upgrade Process Example

```
# Upload and prepare software
upload_result = client.upload_package("/path/to/Unity_5.4.0.bin")
prepare_result = client.prepare_software(upload_result["id"])
candidate_id = prepare_result["id"]

# Verify eligibility before proceeding
# Note: verify_upgrade_eligibility is a stateless endpoint and doesn't require parameters
eligibility = client.verify_upgrade_eligibility()
if not eligibility["eligible"]:
    print("Upgrade not eligible!")
    for message in eligibility["messages"]:
        print(f"- {message}")
    exit(1)

# Create upgrade session
upgrade_session = client.create_upgrade_session(candidate_id, "Planned upgrade to 5.4.0")
session_id = upgrade_session["content"]["id"]

# Monitor upgrade progress
final_status = client.monitor_upgrade_session(session_id, interval=10, timeout=3600)
```

Error Handling

```
from dell_unisphere_client.exceptions import (
    AuthenticationError,
    CSRFTokenError,
    UnisphereClientError
)

try:
    client = UnisphereClient("https://unisphere.example.com", "admin", "wrong_password")
    client.login()
except AuthenticationError as e:
    print(f"Authentication failed: {e}")
except CSRFTokenError as e:
    print(f"Security token error: {e}")
except UnisphereClientError as e:
    print(f"Client error: {e}")
```

CLI Interface

```
# Get system information
$ dell-unisphere-client --host unisphere.example.com --username admin system-info

# Upload a software package
$ dell-unisphere-client --host unisphere.example.com --username admin software upload --file Unity_5.4.0.bin

# Check upgrade eligibility
$ dell-unisphere-client --host unisphere.example.com --username admin upgrade verify --candidate-id candidate_12345

# Create and monitor upgrade
$ dell-unisphere-client --host unisphere.example.com --username admin upgrade create --candidate-id candidate_12345 --monitor
```

Mock Integration

- Seamlessly works with both real Unisphere API and mock backend
- Detects mock environments automatically
- Provides consistent behavior across real and mock environments
- Perfect for testing and development workflows
- Common interface regardless of backend

Test Coverage

- Unit tests for all client components
- Integration tests with mock backend
- Session management tests
- Authentication flow testing
- Error handling and recovery testing
- Full upgrade workflow validation

Implementation Benefits

- Simplified API interaction
- Consistent error handling
- Automatic session management
- Progress monitoring for long-running operations
- Both programmatic and CLI interfaces
- Comprehensive documentation

Real-World Applications

- Automation of storage system upgrades
- Integration with CI/CD pipelines
- Monitoring and reporting tools
- Custom management interfaces
- Backup and disaster recovery solutions

Future Enhancements

- Extended resource type support
- Performance optimization features
- Enhanced logging and diagnostics
- Additional authentication methods
- Configuration validation
- Security scanning integration

Questions?

Thank you for your attention!