NADC Migration Project Summary

Project Overview

Background

The NADC (North American Data Center) migration involves moving infrastructure from RHEL 7 to RHEL 9 as part of a datacenter relocation. The primary challenge is that existing Perl scripts rely heavily on CPAN modules that are not available on the locked-down RHEL 9 environment.

Core Problem

- **Current State**: ControlM jobs invoke Perl scripts that use external CPAN dependencies
- **Target State**: RHEL 9 environment with no external CPAN installation permissions
- **Constraint**: ControlM job definitions must remain unchanged
- **Solution**: Create a hybrid Perl-Python architecture where Python handles CPAN functionality

Strategic Approach

Architecture Decision

Instead of rewriting all Perl scripts or changing ControlM jobs, we implement a "bridge" pattern:

1. ControlM continues to invoke existing Perl scripts (no job changes required)
2. Perl scripts become lightweight wrappers that delegate to Python
3. Python handles the heavy lifting using built-in or easily installable modules
4. The interface remains identical from ControlM's perspective

Key Benefits

- **Zero ControlM Changes**: Job definitions remain unchanged
- **Minimal Code Changes**: Only `use` statements need modification
- **Maintains Compatibility**: Same APIs, return values, and error handling
- **Future Proof**: Easier to maintain and extend Python codebase

CPAN Dependencies Inventory

Database Operations

- **DBI**: Core database interface (Oracle, Informix)
- **Module Status**: Complete replacement implemented

XML Processing

- **XML::Simple**: Basic XML parsing and manipulation
- **XML::XPath**: XPath query support
- **XML::XPath::NodeSet**: XPath node operations
- **Module Status**: Pending implementation

Date/Time Handling

- **Date::Manip**: Complex date calculations
- **Date::Parse**: String-to-date parsing
- **DateTime**: Object-oriented date/time
- **Module Status**: Pending implementation

Web/Network Operations

- **LWP::UserAgent**: HTTP client operations
- **WWW::Mechanize**: Web form automation
- **Net::SFTP::Foreign**: SFTP file transfers
- **Module Status**: Pending implementation

Cryptography & Security

- **Crypt::CBC**: Block cipher operations
- **Crypt::SSLeay**: SSL/TLS support
- **Module Status**: Pending implementation

File Operations

- **Excel::Writer::XLSX**: Excel file generation
- **Module Status**: Pending implementation

Communication

- **Mail::Sender**: Email operations
- **Module Status**: Pending implementation

Utilities

- **Log::Log4perl**: Advanced logging

- **Pod::Find**: Documentation utilities
- **Switch**: Control flow (deprecated in modern Perl)
- **Module Status**: Pending implementation

Technical Implementation

Core Infrastructure Components

CPANBridge.pm

- Base class for all CPAN replacements
- Handles Perl-to-Python communication via JSON over pipes
- Manages error handling, timeouts, and retry logic
- Provides debugging and performance monitoring
- Platform-specific optimizations (Windows vs Linux)

cpan_bridge.py

- Python bridge script that receives and routes requests
- Validates input for security
- Loads and manages helper modules dynamically
- Returns structured JSON responses
- Includes comprehensive error handling and logging

Helper Module Pattern

Each CPAN module replacement follows this structure:

- **Perl Wrapper** (e.g., DBIHelper.pm): Provides identical API to original CPAN module
- **Python Implementation** (e.g., helpers/database.py): Contains actual functionality
- **Bridge Communication**: JSON-based request/response between layers

Communication Flow

```
ControlM Job → Perl Script → CPAN Replacement → CPANBridge → Python Helper → Database/Service
```

Current Status

Completed Components

1. **CPANBridge Infrastructure**: Core communication layer working on both Windows and RHEL 9
2. **DBI Replacement (DBIHelper.pm)**: Complete drop-in replacement for database operations
   - Supports Oracle and Informix connections
   - All standard DBI methods implemented
   - Transaction support included
   - Error handling compatible with existing code

Validated Functionality

- Perl-to-Python communication bridge operational
- JSON serialization/deserialization working
- Windows pipe communication issues resolved
- Basic database connection framework tested
- Error handling and debugging infrastructure functional

Testing Environment

- **Development**: Windows environment for initial development and testing
- **Target**: RHEL 9 production environment
- **Validation**: Bridge functionality confirmed working on both platforms

Next Steps

Phase 1: Database Testing

1. Test DBIHelper with actual Oracle/Informix databases
2. Validate all DBI method compatibility
3. Performance testing and optimization
4. Integration testing with existing Perl scripts

Phase 2: Additional CPAN Modules

Following priority order:

1. XML::Simple (straightforward Python equivalent)
2. Date::Parse (well-supported in Python)

3. Mail::Sender (built-in Python capabilities)
4. Excel::Writer::XLSX (mature Python libraries)
5. WWW::Mechanize (complex but manageable)
6. Remaining modules based on usage frequency

Phase 3: Production Deployment

1. RHEL 9 environment setup and testing
2. Python dependency installation (oracledb, etc.)
3. Staged migration of ControlM jobs
4. Monitoring and performance validation

Risk Mitigation

Technical Risks

- **Performance Impact**: JSON serialization adds ~2-4ms per operation (acceptable vs database query times)
- **Compatibility Issues**: Extensive testing planned for each CPAN replacement
- **Platform Differences**: Windows development environment validated against RHEL 9 target

Operational Risks

- **Migration Complexity**: Phased approach reduces risk
- **Rollback Strategy**: Original RHEL 7 environment maintained during transition
- **Testing Coverage**: Comprehensive testing planned for each component

Success Metrics

Technical Metrics

- Zero ControlM job definition changes required
- <5ms performance overhead per operation
- 100% API compatibility for replaced CPAN modules
- Zero data integrity issues during migration

Business Metrics

- On-time datacenter migration completion
- No business process interruption

- Reduced maintenance complexity post-migration
- Future-ready architecture for additional enhancements

Architecture Benefits

Maintainability

- Clear separation of concerns between Perl and Python layers
- Python codebase easier to maintain than CPAN dependencies
- Centralized error handling and logging
- Standardized communication protocol

Scalability

- Easy to add new CPAN module replacements
- Bridge infrastructure supports any number of helper modules
- Performance monitoring built-in for optimization
- Platform-agnostic design supports future environments

Security

- Input validation at Python bridge layer
- No direct system access from Perl scripts
- Centralized dependency management
- Reduced attack surface vs full CPAN installation