==========

==========

CATEGORY: PowerShell

==========

==========

PowerShell is the backbone of Windows automation and administrative control. These SOPs ensure every PowerShell-related action performed through RDAM Script Wizard is **controlled**, **auditable**, and aligned with **enterprise security and operational standards**.

SOP 1 – Run PowerShell Script

Script Name: Run PowerShell Script **Category:** PowerShell **Version:** 1.0 **Approved By:** IT Operations / Security

1. Purpose

This script executes a PowerShell script block or file in a controlled, logged manner. It supports automation, troubleshooting, and administrative workflows while ensuring traceability and safety.

2. Scope

- Windows servers and workstations
- PowerShell 5.1 and PowerShell 7+
- Used by system administrators, engineers, and automation teams

3. Definitions

- **Script Block:** Inline PowerShell code.
- **Execution Policy:** Security mechanism controlling script execution.

4. Preconditions

- Operator must have permissions required by the script content.
- Execution policy must allow script execution (or be bypassed safely).
- Script must be validated and approved if performing sensitive actions.

5. Required Inputs

- Script block or script file path
- Optional: Parameters
- Optional: Execution policy override

6. Procedure Steps

1. **Input Collection**
    - Wizard prompts for script content or file path.
    - Validate script is not empty.
2. **Safety Validation**
    - Check for disallowed commands (if policy enforced).
    - Confirm operator authorization for sensitive operations.
3. **Execution Policy Handling**
    - If override selected, apply temporary bypass.
4. **Script Execution**
    - Execute script block or file.
    - Capture output, warnings, and errors.
5. **Output Formatting**
    - Present structured results.
6. **Logging**
    - Log script hash, operator, timestamp, and result status.

7. Expected Output

- Script output, including success or error details.

8. Post-Execution Validation

- Operator may verify system state changes manually.

9. Error Handling

- Syntax errors
- Access denied
- Execution policy restrictions

- Missing file

10. Security Considerations

- PowerShell can modify system state; strict controls required.
- Script content must not be logged verbatim unless approved.

11. Audit Logging Requirements

- Operator ID
- Script hash
- Parameters
- Timestamp
- Success/Failure

12. Organizational Benefit Statement

This script provides a safe, auditable method for executing PowerShell code, supporting automation and troubleshooting while maintaining governance.

SOP 2 – Get PowerShell Version

Script Name: Get PowerShell Version **Category:** PowerShell

1. Purpose

This script retrieves the installed PowerShell version(s) on the system, supporting compatibility checks, troubleshooting, and environment validation.

2. Scope

- PowerShell 5.1
- PowerShell 7+ (if installed)

3. Definitions

- **PSVersionTable:** Built-in variable containing version metadata.

4. Preconditions

- Operator must have permission to run PowerShell commands.

5. Required Inputs

- None

6. Procedure Steps

1. **Initialize Query**
    - Retrieve `$PSVersionTable`.
2. **Version Detection**
    - Identify:
        - Major/minor version
        - Edition (Desktop/Core)
        - CLR version
        - Build version
3. **Output Formatting**
    - Present structured version information.
4. **Logging**
    - Log operator and timestamp.

7. Expected Output

- PowerShell version details.

8. Post-Execution Validation

- Operator may verify using `$PSVersionTable` manually.

9. Error Handling

- PowerShell not installed (rare)
- Access denied

10. Security Considerations

- None beyond standard PowerShell execution controls.

11. Audit Logging Requirements

- Operator ID
- Timestamp

12. Organizational Benefit Statement

This script provides a reliable, auditable method for retrieving PowerShell version information, supporting compatibility and troubleshooting workflows.

SOP 3 – List Installed PowerShell Modules

Script Name: List Installed PowerShell Modules **Category:** PowerShell

1. Purpose

This script enumerates installed PowerShell modules, supporting troubleshooting, dependency validation, and environment auditing.

2. Scope

- Modules installed for current user or all users
- PowerShell 5.1 and 7+

3. Definitions

- **Module:** A package containing PowerShell functions, cmdlets, or providers.

4. Preconditions

- Operator must have permission to query module directories.

5. Required Inputs

- Optional: Module name filter

6. Procedure Steps

1. **Input Collection**
    - Wizard prompts for optional filter.
2. **Module Enumeration**
    - Retrieve modules from:
        - System module paths
        - User module paths
3. **Filtering**
    - Apply name filter if provided.
4. **Output Formatting**

- Present:
    - Module name
    - Version
    - Path
    - Exported commands

5. **Logging**
    - Log filter, operator, timestamp.

7. Expected Output

- List of installed modules.

8. Post-Execution Validation

- Operator may verify using `Get-Module -ListAvailable`.

9. Error Handling

- Access denied
- Invalid filter

10. Security Considerations

- Modules may contain sensitive scripts; restrict access.

11. Audit Logging Requirements

- Operator ID
- Filter used
- Timestamp

12. Organizational Benefit Statement

This script provides a controlled, auditable method for reviewing installed PowerShell modules, supporting troubleshooting and governance.

SOP 4 – Install PowerShell Module

Script Name: Install PowerShell Module **Category:** PowerShell

1. Purpose

This script installs a PowerShell module from a repository (e.g., PSGallery), supporting automation, development, and operational workflows.

2. Scope

- PowerShell 5.1 and 7+
- Online or internal repositories

3. Definitions

- **Repository:** Source of PowerShell modules (e.g., PSGallery).
- **Module Installation:** Downloading and registering module on system.

4. Preconditions

- Operator must have administrative rights (for all-users install).
- Repository must be reachable.
- Module installation must be authorized.

5. Required Inputs

- Module name
- Optional: Version
- Optional: Scope (CurrentUser/AllUsers)

6. Procedure Steps

1. **Input Collection**
    - Wizard prompts for module name and options.
2. **Repository Validation**
    - Confirm repository availability.
3. **Module Lookup**
    - Search repository for module.
    - If version specified, validate availability.
4. **Installation Operation**
    - Install module with selected scope.
5. **Post-Install Verification**

    - Confirm module appears in `Get-Module -ListAvailable`.
6. **Logging**
    - Log module name, version, scope, operator, timestamp.

7. Expected Output

- Confirmation of successful module installation.

8. Post-Execution Validation

- Operator may import module manually.

9. Error Handling

- Module not found
- Repository unreachable
- Access denied
- Version mismatch

10. Security Considerations

- Installing modules may introduce untrusted code; ensure repository trust.
- Use internal repositories when possible.

11. Audit Logging Requirements

- Operator ID
- Module name
- Version
- Scope
- Timestamp

12. Organizational Benefit Statement

This script ensures module installation is performed safely and with full accountability, supporting automation and operational consistency.