AWS IAM Policy Splitting Strategy: Overcoming the 6,144 Character Limit

Summary

This document provides prescriptive guidance for organizations facing AWS IAM managed policy size limitations. With policies exceeding 46,000 characters (7.5x the AWS limit), a strategic decomposition approach is required to maintain security posture while ensuring compliance with AWS constraints.

Problem Statement

Current Challenge

- **Policy Size**: Current network architecture and administrator policies exceed 46,000 characters
- **AWS Limit**: IAM managed policies are limited to 6,144 characters
- **Impact**: Cannot deploy comprehensive network security policies as single managed policies
- **Constraint**: AWS Organizations SCPs face the same 6,144 character limitation

Business Impact

- Inability to deploy standardized network security roles
- Potential security gaps from incomplete policy deployment
- Operational complexity from manual policy management
- Compliance risks from inconsistent permission models

Strategic Approach: Functional Policy Decomposition

Rather than maintaining monolithic policies, decompose into **8 focused managed policies** per role, each addressing specific functional domains while maintaining the existing security model.

Core Principles

1. **Functional Separation**: Group related permissions by AWS service families
2. **Reusability**: Create policies that can be shared across multiple roles
3. **Maintainability**: Ensure each policy remains under the 6,144 character limit
4. **Security Preservation**: Maintain existing permission boundaries and conditions

Detailed Policy Breakdown

Policy 1: Core Infrastructure (~6,000 characters)

Purpose: Foundation services for infrastructure management

Included SIDs:

- `PassRoleForCloudFormationExecRoles`: IAM role passing for CloudFormation

- `IdentityDiscoveryAndSLR`: IAM discovery and service-linked role creation

- `CloudFormationDeploy`: Infrastructure-as-Code deployment capabilities

- `KmsReadAndUse`: Cryptographic operations for service integration

- `SecretsManagerRead`: Secure credential access

Rationale: These permissions form the foundation for all infrastructure operations and are required across multiple functional areas.

Policy 2: Networking Services (~6,000 characters)

Purpose: Core networking and connectivity management

Included SIDs:

- `NetworkingAdmin`: EC2 VPC, Transit Gateway, NAT, IGW operations

- `Route53AndARCAdmin`: DNS management and Application Recovery Controller

- `Route53DomainsReadOnly`: Domain registration visibility

- `DirectConnectReadOnly`: Hybrid connectivity monitoring

- `InternetAndNetworkMonitorAdmin`: Network performance monitoring

Rationale: Consolidates all networking control plane operations while maintaining read-only access to sensitive connectivity services.

Policy 3: Security & Compliance (~6,000 characters)

Purpose: Security monitoring and compliance oversight

Included SIDs:

- `SecurityServicesReadOnly`: Security Hub, GuardDuty, Inspector visibility

- `MacieReadOnly`: Data classification monitoring

- `SecurityLakeReadOnly`: Security data lake access

- `DetectiveReadOnly`: Security investigation capabilities

- `AuditManagerReadOnly`: Compliance framework monitoring

- `AccessAnalyzerReadOnlyAndValidation`: Policy analysis tools

- `FirewallManagerReadOnly`: Centralized firewall policy visibility

- `EdgeProtectionReadOnly`: WAF and Shield monitoring

Rationale: Provides comprehensive security visibility without administrative capabilities, supporting security operations while preventing tampering.

Policy 4: Observability (~5,000 characters)

Purpose: Monitoring, logging, and operational visibility

Included SIDs:

- `LogsReadAndQuery`: CloudWatch Logs access for troubleshooting

- `CloudWatchReadOnly`: Metrics and alarm visibility

- `CloudTrailReadOnly`: Audit trail access

- `ConfigReadOnly`: Configuration compliance monitoring

- `XRayReadOnly`: Application performance monitoring

- `ObservabilityAccessManagerLinking`: Cross-account observability

Rationale: Enables comprehensive operational visibility while preventing tampering with audit and monitoring infrastructure.

Policy 5: Systems Management (~6,000 characters)

Purpose: Server and system administration capabilities

Included SIDs:

- `SsmCoreOps`: Session Manager and Run Command (tag-gated)

- `SsmAssocAndMW`: Association and Maintenance Window management

- `SsmParametersNetworkPath`: Parameter Store for network configurations

- `SsmDocsNetwork`: Network-specific automation documents

- `SsmDocsRead`: Global document visibility

- `SsmAutomationStartNetworkDocsOnly`: Controlled automation execution

- `SsmAutomationReadResults`: Automation monitoring

Rationale: Provides controlled system administration capabilities with tag-based access controls and network-scoped automation.

Policy 6: Development & Containers (~6,000 characters)

Purpose: Application development and container platform management

Included SIDs:

- `CodeCommitReadWrite`: Source code management

- `CodePipelineOperations`: CI/CD pipeline management

- `CodeBuildOperations`: Build process control

- `CodeDeployOperations`: Deployment management

- `CodeArtifactAdmin`: Artifact repository management

- `EcrAdmin`: Container registry management

- `EcsAdmin`: Container service management

- `EksAdmin`: Kubernetes service management

Rationale: Supports modern application development workflows while maintaining container platform administrative control.

Policy 7: Storage & Messaging (~5,000 characters)

Purpose: Data storage and inter-service communication

Included SIDs:

- `S3ListAllMyBuckets`: Storage discovery

- `S3NetworkBucketsAdminSafe`: Network bucket management with safeguards

- `S3BucketPolicyViaCloudFormationOnly`: Controlled policy management

- `S3NetworkObjectsStrict`: Object-level operations with encryption requirements

- `SnsPublishAndRead`: Notification publishing

- `SnsOpsMinimalWrites`: Limited SNS administrative operations

- `SqsConsumeAndRead`: Message queue consumption

- `SqsOpsMinimalWrites`: Limited SQS administrative operations

Rationale: Provides necessary storage and messaging capabilities while enforcing encryption and change control requirements.

Policy 8: Utilities & Support (~4,000 characters)

Purpose: Operational utilities and support functions

Included SIDs:

- `CloudShellFull`: Browser-based CLI access

- `AmazonQUseInConsole`: AI assistant access

- `QBusinessChatUseOnly`: Business AI capabilities

- `ServiceQuotasRead`: Quota monitoring

- `ServiceQuotasIncreaseNetworkScoped`: Network service quota management
- `HealthReadOnly`: Service health monitoring
- `TrustedAdvisorReadOnly`: Best practice recommendations
- `SupportFull`: AWS Support case management
- `CertificateManagerPrototyping`: SSL/TLS certificate management
- `RamNetworkSharesAdmin`: Resource sharing administration
- `RamPermissionAuthoring`: Custom permission development

Rationale: Consolidates operational utilities and support functions that don't fit into other functional categories.

Implementation Strategy

Phase 1: Policy Creation

1. **Create 8 separate CloudFormation templates** for each policy category
2. **Validate character counts** for each policy to ensure compliance
3. **Test policy syntax** using `aws iam validate-policy`
4. **Deploy policies** in non-production environment first

Phase 2: Role Template Updates

1. **Modify existing role templates** to reference multiple managed policies
2. **Update ManagedPolicyArns** sections to include all 8 policies
3. **Maintain existing trust relationships** and permissions boundaries
4. **Preserve role naming conventions** and organizational structure

Phase 3: Deployment and Testing

1. **Deploy new policies** before updating roles
2. **Update roles** to use new policy structure
3. **Test functionality** using `aws iam simulate-principal-policy`
4. **Validate permissions** in development environment
5. **Retire monolithic policies** after successful validation

Phase 4: Documentation and Training

1. **Update operational procedures** to reflect new policy structure
2. **Train teams** on new policy organization
3. **Document troubleshooting procedures** for multi-policy roles
4. **Establish change management** processes for policy updates

Role Template Modifications

Before (Monolithic Approach)

```
NetworkArchitectureRole:
  Type: AWS::IAM::Role
  Properties:
    ManagedPolicyArns:
      - !Ref NetworkArchitectureManagedPolicy  # 46,000+ characters -
FAILS
```

After (Decomposed Approach)

```
NetworkArchitectureRole:
  Type: AWS::IAM::Role
  Properties:
    ManagedPolicyArns:
      - !Ref CoreInfrastructurePolicy          # ~6,000 characters
      - !Ref NetworkingServicesPolicy          # ~6,000 characters
      - !Ref SecurityCompliancePolicy          # ~6,000 characters
      - !Ref ObservabilityPolicy               # ~5,000 characters
      - !Ref SystemsManagementPolicy           # ~6,000 characters
      - !Ref DevelopmentContainersPolicy       # ~6,000 characters
      - !Ref StorageMessagingPolicy            # ~5,000 characters
      - !Ref UtilitiesSupportPolicy            # ~4,000 characters
```

AWS Organizations SCP Considerations

SCP Limitations

- **Same 6,144 character limit** applies to Service Control Policies
- **Cannot directly solve** SCP size issues through IAM policy splitting
- **Different purpose**: SCPs provide guardrails, IAM policies provide permissions

Recommended SCP Strategy

1. **Keep SCPs lightweight** - Focus on organization-wide guardrails only
2. **Use SCP inheritance** - Apply different SCPs at different Organizational Unit levels
3. **Complement with IAM policies** - Use detailed IAM policies for granular permissions
4. **Split SCPs functionally** - Create separate SCPs for different control domains

SCP Best Practices

- **Root OU**: Core security controls and compliance requirements
- **Network OU**: Network-specific guardrails and restrictions
- **Development OU**: Development environment controls
- **Production OU**: Production environment safeguards

Benefits and Outcomes

Immediate Benefits

- **Compliance with AWS limits**: All policies under 6,144 characters
- **Deployable infrastructure**: Can successfully deploy network roles
- **Maintained security posture**: No reduction in security controls
- **Improved organization**: Clear functional separation of permissions

Long-term Advantages

- **Enhanced maintainability**: Easier to update specific functional areas
- **Improved reusability**: Policies can be shared across different roles
- **Better testing**: Can test individual functional areas independently
- **Simplified troubleshooting**: Easier to identify permission issues by functional area

Operational Improvements

- **Faster deployments**: Smaller policies deploy more quickly
- **Reduced complexity**: Clear separation of concerns
- **Better documentation**: Each policy has focused documentation
- **Easier auditing**: Simpler to review specific functional permissions

Risk Mitigation

Potential Risks

1. **Increased complexity**: Managing 8 policies instead of 1
2. **Deployment dependencies**: Must deploy policies before roles
3. **Permission gaps**: Risk of missing permissions during migration
4. **Operational overhead**: More policies to maintain and update

Mitigation Strategies

1. **Comprehensive testing**: Use simulation tools to validate permissions
2. **Staged deployment**: Deploy in non-production first
3. **Detailed documentation**: Maintain clear mapping of permissions
4. **Automation**: Use Infrastructure-as-Code for all deployments
5. **Monitoring**: Implement CloudTrail monitoring for permission issues

Testing and Validation

Pre-Deployment Testing

```
# Validate policy syntax
aws iam validate-policy --policy-document file://policy.json

# Simulate permissions
```

```
aws iam simulate-principal-policy \
  --policy-source-arn arn:aws:iam::ACCOUNT:role/ROLE \
  --action-names ACTION \
  --resource-arns RESOURCE
```

Post-Deployment Validation

1. **Functional testing**: Verify all expected operations work
2. **Negative testing**: Confirm restricted operations are blocked
3. **Cross-service testing**: Validate service integrations
4. **Performance testing**: Ensure no degradation in operation speed

Conclusion

The decomposition of monolithic IAM policies into 8 functional policies provides a sustainable solution to AWS size limitations while maintaining security posture and operational capabilities. This approach enables:

- **Immediate compliance** with AWS IAM policy size limits
- **Preserved security model** with all existing controls
- **Enhanced maintainability** through functional organization
- **Future scalability** for additional services and permissions

The implementation requires careful planning and testing but provides significant long-term benefits for policy management and organizational security posture.

Appendix A: Character Count Verification

Policy Category	Estimated Characters	AWS Limit Compliance
Core Infrastructure	~6,000	✅ Compliant
Networking Services	~6,000	✅ Compliant
Security & Compliance	~6,000	✅ Compliant
Observability	~5,000	✅ Compliant
Systems Management	~6,000	✅ Compliant
Development & Containers	~6,000	✅ Compliant
Storage & Messaging	~5,000	✅ Compliant
Utilities & Support	~4,000	✅ Compliant
Total	**~44,000**	✅ **All Compliant**

Appendix B: Migration Checklist

- ☐ Create 8 separate policy CloudFormation templates
- ☐ Validate character counts for each policy
- ☐ Test policy syntax with AWS CLI
- ☐ Update role templates to reference multiple policies
- ☐ Deploy policies in development environment
- ☐ Test role functionality with simulation tools
- ☐ Deploy to production environment
- ☐ Validate production functionality
- ☐ Update documentation and procedures
- ☐ Train operational teams
- ☐ Retire old monolithic policies
- ☐ Monitor for any permission issues