Proxylity UDP Gateway
Solution Overview

Version: 1.4 | June 2026

available in aws marketplace

aws PARTNER

aws Qualified Software

Executive Summary

Proxylity UDP Gateway is a fully managed service that eliminates the overhead of maintaining infrastructure servicing UDP workloads. By connecting UDP traffic directly to AWS serverless resources Proxylity UDP Gateway enables organizations to modernize their network applications and dramatically increase development team velocity while reducing costs, complexity, and security risks.

What Proxylity Does

Proxylity acts as a global, scalable and reliable "front door" for applications using UDP for edge to cloud communication by handling all low-level traffic management and then batching and forwarding packet data to backend serverless resources in your AWS account. This creates a logical separation between the network infrastructure and backend business logic.

Instead of endless cycles of deploying, customizing, updating and managing servers, containers and VPCs to handle UDP traffic, organizations focus on the tasks and information represented by packets with well-established Event Driven Architecture and serverless architecture patterns using UDP Gateway's seamless integration with AWS.

Batches of packets may be efficiently and economically handled with Lambda functions and simultaneously directed to other services like S3, DynamoDB, or Firehose (and more). When combined with Proxylity's "IaC First" principals, this decoupling of backend systems improves change delivery time metrics.

Key Value Proposition

- **Accelerate Development**: Faster development and deployment cycles deliver results for customers sooner
- **Eliminate Server Management**: No more patching, updating, or maintaining VMs/containers for UDP services
- **Reduce Operational Costs**: Pay only for packets processed, not idle infrastructure. Enable per-developer and per-branch environments without overhead
- **Enhance Security**: Integrated with AWS IAM and least-privilege permissions. Built-in firewall capabilities
- **Achieve Global Scale**: Deploy across multiple regions with automatic scaling, automatic failover and high availability

 Document Version: 1.4 | Date: June 2026

Enterprise Features & Benefits

Accelerate Development

Faster development and deployment cycles deliver results for customers sooner

Today, building UDP-based systems typically forces teams into an infrastructure-first model. Even when the application logic is simple—device telemetry, signaling, or control messages—teams must first stand up UDP-capable infrastructure using EC2 fleets, containers, or Kubernetes services. In many cases, organizations default to higher-level protocols like MQTT or TCP-based APIs simply because the operational burden of raw UDP is assumed to be too high.

This shifts focus away from application behavior and into infrastructure design, networking configuration, and long-lived service management. As a result, product iteration slows down, and experimentation is constrained by how quickly infrastructure can be provisioned and safely modified.

Proxylity reverses this model by enabling an IaC-first UDP service definition. A team starts with a simple deployment template that defines a Listener and a destination. Within minutes, packets are flowing into backend processing pipelines without requiring any custom networking infrastructure or long-lived servers. From there, development proceeds through rapid, low-risk iterations where new destinations are added or modified via standard infrastructure-as-code workflows.

Because deployments typically complete in tens of seconds to a few minutes, teams can iterate continuously without waiting on infrastructure coordination or release windows. A single UDP ingress point can be extended over time into multiple backend destinations, including Lambda processing pipelines, Step Functions workflows, or data sinks such as S3, DynamoDB, and Firehose.

A key differentiator is architectural evolution without protocol change. Devices continue sending the same lightweight UDP packets while backend systems evolve independently. Through multiplexing, a single stream of packets can simultaneously support multiple downstream systems—operational processing, analytics pipelines, and integration layers—without modifying edge behavior. This includes patterns such as “inside-out API Gateway,” where UDP ingress feeds existing HTTP-based services in the backend.

The result is a shift in the development model: UDP services are no longer infrastructure projects. They become standard application systems that evolve through familiar cloud-native workflows, with rapid iteration cycles and decoupled

backend design.

Eliminate Server Management

No more patching, updating, or maintaining VMs and containers for UDP services

In traditional systems, services are commonly implemented as long-lived processes running on virtual machines or containers. Examples include RADIUS servers, telemetry collectors, or custom daemons that require persistent runtime environments. These systems often include custom extensions, configuration files, and runtime state that require careful coordination during updates.

Operationally, this introduces continuous maintenance overhead. Configuration changes often require process restarts or rolling deployments across fleets. Even when hot-reload mechanisms exist, they tend to be fragile or inconsistent across environments. As systems scale, maintaining version consistency across nodes becomes increasingly difficult, and upgrades introduce risk of partial rollout states and service disruption.

Scaling introduces additional complexity. Teams often rely on static fleets sized for peak traffic, or orchestration layers such as Kubernetes that introduce their own networking and lifecycle complexity. In many cases, services are paired with TLS termination clusters or edge proxies that provide no application differentiation but add significant operational burden.

Release processes are particularly fragile. Because many services maintain long-lived connections or session-like behavior, deployments require careful draining strategies to avoid service interruption. Even with controlled rollouts, connection churn can still result in transient failures during upgrades.

Incident recovery is another major pain point. Under load or failure conditions, connection-heavy systems can experience “pinning” effects where active sessions or handshake storms prevent nodes from recovering cleanly. This can lead to cascading degradation or prolonged recovery times during traffic spikes or partial outages.

Proxylity eliminates these entire classes of operational complexity by removing the need for customer-managed servers altogether. There are no long-lived ingress services to patch, scale, or upgrade. There are no connection-draining strategies, because there are no persistent connections at the ingress layer.

Instead, Proxylity acts as a managed, stateless ingestion layer that batches and forwards packet data into serverless backends. Traffic variability, including sudden spikes or “thundering herds,” is absorbed at the ingress layer and delivered as

discrete event batches to downstream systems. This ensures that backend services remain focused on processing business logic rather than managing network-level behavior.

As a result, operational responsibility shifts away from infrastructure lifecycle management and toward application-level event processing using standard AWS services.

Reduce Operational Costs

Pay only for packets processed, not idle infrastructure

A major cost inefficiency in traditional systems is the requirement for always-on infrastructure. Even when traffic is intermittent or bursty, teams must maintain continuously running server fleets, load balancers, and supporting systems. This includes both raw compute costs and the operational overhead required to keep those systems healthy and available.

This model becomes especially inefficient in systems such as telemetry ingestion, device communication, or control-plane messaging, where traffic is often highly variable. Infrastructure is typically provisioned for peak load and remains underutilized for extended periods. Even “serverless” adjacent systems, such as managed streaming platforms, often impose significant baseline costs that limit the number of environments teams can afford to operate.

This creates a structural constraint: organizations reduce the number of deployed stacks, limit per-developer or per-feature environments, and carefully ration infrastructure usage. As a result, experimentation slows down, and development workflows become gated by infrastructure cost rather than engineering need.

Proxylity removes this constraint by eliminating the need for always-on UDP infrastructure. Costs are directly tied to packet ingestion and processing rather than idle capacity. Because backend processing is delegated to serverless services such as Lambda, SQS, DynamoDB, and Firehose, compute resources scale dynamically with demand and do not require provisioning or idle capacity planning.

This enables a fundamentally different operating model where teams can spin up ephemeral environments freely. Per-developer or per-branch UDP stacks become practical because there is no fixed infrastructure footprint to maintain. This directly accelerates development velocity, especially in environments that increasingly rely on AI-assisted development and rapid iteration.

The cost model also enables broader architectural adoption of multi-region systems. While many organizations avoid active-active or multi-region deployments due to

cost and complexity overhead, Proxylity provides globally distributed ingress infrastructure by default. Customers can choose to deploy single-region systems or extend to multi-region architectures without duplicating full infrastructure stacks.

This reduces both direct infrastructure spend and indirect organizational cost, including the engineering time required to manage and coordinate always-on systems.

Enhance Security

Integrated with AWS IAM and least-privilege permissions with built-in ingress controls

Security in traditional systems is typically handled at the transport layer using mechanisms such as mTLS, DTLS, or VPN-based connectivity. These approaches introduce significant complexity, including certificate lifecycle management, PKI infrastructure, and protocol-level security configuration. In many cases, these systems are over-engineered relative to the actual security requirements of the application.

Simpler approaches, such as IP allowlists or security groups, are often used in practice but are difficult to manage consistently across environments and regions. As systems scale, security configuration becomes fragmented across infrastructure layers, application logic, and network controls.

Proxylity simplifies this model by providing flexible ingress security controls combined with AWS-native identity and access management. At the transport layer, Proxylity supports WireGuard-based listeners that can operate in either strict peer-identity mode or more flexible “server-validated” modes. This allows customers to choose between explicit device identity enforcement or simpler connectivity models depending on their requirements.

Additional controls such as IP restrictions and listener-level filtering provide further granularity over who can send traffic to a given endpoint. This enables security policies to be enforced centrally at the ingress layer rather than duplicated across multiple backend systems.

At the cloud integration layer, Proxylity uses AWS IAM to deliver packets directly into customer-owned resources. Data is not retained in the Proxylity control plane; instead, it is forwarded into customer AWS accounts under their own permission boundaries. Customers retain full control over access policies using familiar AWS tools and constructs, without introducing new security paradigms or credential systems.

This results in a security model that is both simpler and more consistent. Rather than distributing security logic across transport layers, application code, and infrastructure configurations, security is centralized at ingress and unified with existing AWS identity and access controls.

Achieve Global Scale

Deploy across multiple regions with automatic scaling, automatic failover, and high availability

Global UDP systems traditionally require significant engineering effort to scale across regions. Teams must design custom routing strategies, deploy parallel infrastructure stacks per region, and implement failover mechanisms that account for connection state and session continuity. These systems are typically complex, expensive, and rarely fully active-active in practice due to operational overhead.

Proxylity simplifies global scaling by providing Anycast-based ingress endpoints that are globally distributed. Client packets automatically route to the nearest available ingress region, minimizing latency from the edge to the network entry point. Once ingested, Proxylity evaluates configured destination policies to route packets to the optimal backend region, whether local or geographically closest.

This design optimizes latency in two dimensions: first by minimizing client-to-ingress distance using Anycast routing, and second by minimizing ingress-to-backend distance using region-aware delivery over AWS's high-performance backbone network.

Global resilience is built into the ingress layer. If a region becomes unavailable, traffic is automatically rerouted to the next closest available ingress region. Because UDP is stateless at the transport level, there are no connections to migrate, drain, or recover. Failover becomes instantaneous from the perspective of the client device.

This architecture enables active-active global deployments as a configuration choice rather than an architectural undertaking. Organizations can deploy single-region systems when appropriate, or extend to multi-region active-active systems without duplicating full networking infrastructure or implementing custom failover logic.

The result is a globally resilient UDP ingestion system where scaling, routing, and failover are handled by the platform, while backend systems remain focused on application logic within standard AWS services.

Operational Excellence

- **Zero Infrastructure Management**: Fully managed service eliminates

server maintenance, patching, and scaling concerns
- **Global Multi-Region Architecture**: Deployed across multiple AWS regions for highest availability and lowest latency
- **Automatic Scaling**: Handles traffic spikes automatically without manual intervention
- **Comprehensive Monitoring**: Integration with AWS CloudWatch Logs and Metrics for full observability

Cost Optimization

- **Consumption-Based Pricing**: Pay only for packets processed, not idle infrastructure
- **Intelligent Batching**: Reduces AWS API calls by up to 90% through asynchronous packet batching
- **Tiered Pricing**: Progressive discounts as volume grows, with batch pricing for enterprise workloads
- **No Upfront Costs**: Start free with 1M packets/month, scale as needed

Security & Compliance

- **AWS IAM Integration**: Leverage existing AWS security model with least-privilege permissions
- **Flexible Firewall**: IP/CIDR-based and policy-based client allow-listing
- **Data Sovereignty**: Your data stays in your AWS account - Proxylity doesn't store or inspect traffic
- **Audit Trail**: Complete transparency with detailed logging and billing information

Technical Capabilities

- **Comprehensive AWS Integration**: Supports Lambda, Step Functions, S3, Firehose, DynamoDB, SNS, SQS, CloudWatch
- **JSON Packet Format**: Standardized data format with metadata for easy processing
- **Multi-Region Destinations**: Deliver packets to AWS resources across multiple regions
- **Configurable Batching**: Flexible per-destination batching with concurrent processing
- **High Performance**: Optimized for low-latency and high-throughput UDP traffic

Diverse Use Cases

- **Modernize Enterprise Services**: Replace archaic SYSLOG, RADIUS, DNS

and other implementations

- **Greenfield Applications**: Support custom UDP protocols and proprietary network and application services
- **Remote Sensing and Telemetry**: Flexible and inexpensive processing of remote sensor data from low to global scale
- **IoT Communication**: Process UDP messages from IoT devices at global scale
- **Gaming Infrastructure**: Handle high-volume, low-latency UDP traffic for game servers

Plan Features

Enterprise Plan - $850/month ($10,200/year)

- **Unlimited Listeners** (no per-listener charges, subject to account limit)
- **Batch-based pricing** starting at $2.50/M batches for significant savings on high-volume workloads
- **Multi-AWS account support**
- **Higher availability SLA**
- **Priority support**
- **Dedicated account management**

Pay as You Go Plan - (Free Tier plus Usage)

- **1 Listener included** (optional hard limit)
- **1M packets/month included** (optional hard limit)
- **Packet-based pricing** starts at $1.25/M packets
- **AWS Marketplace subscription** required
- **Online documentation**
- **Community support**

Dedicated Stacks - Custom Pricing

- Single-tenant infrastructure
- Configurable regionality
- Large numbers of ports with contiguous ranges
- Custom SLAs and compliance processes
- Contact sales for more information
- Custom Integration available

Technical Integration

Integrated AWS Services

- **Security**: IAM, KMS
- **Observability**: CloudWatch Logs and Metrics
- **Provisioning**: AWS Marketplace
- **CI/CD**: CloudFormation and AWS Serverless Application Model

Supported AWS Destination Services

- **Compute**: Lambda, Step Functions
- **API & Integration**: API Gateway, EventBridge, IoT Core
- **Storage & Data**: DynamoDB, S3
- **Streaming & Processing**: Kinesis Firehose, Kinesis Data Streams
- **Messaging & Monitoring**: SNS, SQS, CloudWatch Logs

Infrastructure as Code Support

- **CloudFormation** custom resources
- **Terraform** modules
- IAM role and policy integration
- Automated deployment capabilities
- CI/CD pipeline integration

Technical Onboarding Process

1. Subscribe via AWS Marketplace
2. Configure IAM for developer and pipeline access
3. Deploy and learn from example solutions on GitHub
4. Develop custom business solutions and workflows
5. Economically test with environment-specific or other fine-grained stacks
6. Deploy production workloads

Why Choose Proxylity?

Proven Track Record

- Available in AWS Marketplace
- AWS Partner Network member with Qualified Software
- Designed for enterprise-grade reliability and scale
- > 99.99% Availability Nov 2024 through June 2026

Developer-Friendly

- First-class Infrastructure as Code
- Comprehensive documentation and examples
- GitHub repository with sample implementations
- Community and enterprise support options

Business Value

- **Faster Time-to-Market**: Eliminate infrastructure setup and management
- **Reduced TCO**: Lower operational costs and resource requirements
- **Enhanced Agility**: Enable rapid development and deployment cycles
- **Lower Risk**: Reduce project delay, security vulnerabilities and compliance concerns

Next Steps

Ready to modernize your UDP infrastructure and eliminate operational overhead? Subscription is always available on AWS Marketplace.

Contact Information:

- **Sales Inquiries**: sales@proxylity.com
- **Technical Support**: Available through AWS Marketplace
- **Documentation**: https://proxylity.com/docs/
- **GitHub Examples**: https://github.com/proxylity/examples

Get Started:

1. Subscribe via AWS Marketplace
2. Start with the Free plan (1M packets/month)
3. Explore documentation and examples
4. Schedule a technical consultation for a PoC
5. Build and expand

Legal Information

Terms of Service Summary

By using Proxylity UDP Gateway, you agree to:

- **Account Responsibility**: Maintain confidentiality of your account credentials and ensure accurate registration information
- **AWS Integration**: Connect your Proxylity account with your AWS account via AWS Marketplace, with charges appearing on your AWS bill
- **Compliance**: Use the service only for lawful purposes and comply with all applicable US laws and regulations
- **Data Security**: Not transmit malicious code or use the service for unauthorized purposes

Privacy Policy Summary

Proxylity is committed to protecting your privacy:

- **Data Collection**: We collect only necessary account information (name, email, AWS account details) and basic usage data
- **Data Use**: Information is used solely to provide and improve our services
- **Data Protection**: Your network traffic data stays in your AWS account - Proxylity doesn't store or inspect packet contents
- **Your Rights**: You have the right to access, correct, or delete your personal data at any time
- **Legal Basis**: Processing is based on contract performance, legal compliance, and legitimate business interests

For complete terms and privacy details, visit:

- **Full Terms of Service**: https://proxylity.com/terms-of-service.html
- **Complete Privacy Policy**:https://proxylity.com/privacy-policy.html

Compliance & Security

- **Data Sovereignty**: All customer data remains in customer's AWS account, and regions configured for destinations
- **AWS Partner Network**: Certified AWS partner with enterprise-grade security standards
- **Audit Trail**: Transparent billing with detailed logging