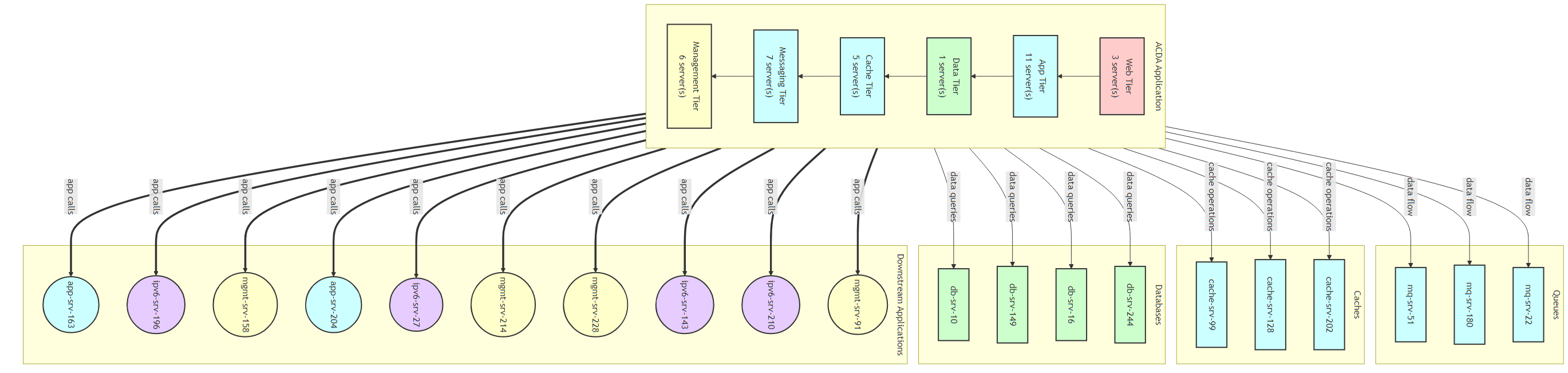
Network Segmentation  
Solutions Architecture Document - ACDA

**Version: 1.0**Date: October 13, 2025  
Classification: Internal

Prepared by: Network Security Team  
Auto-generated by Network Segmentation Analyzer

# Application Data Flow Diagram

The following diagram shows the architecture and data flows for ACDA, including internal application tiers, downstream applications, and infrastructure dependencies.



## Diagram Legend

* Application Box (with direction TB) = Internal architecture showing application tiers
* Web Tier = Frontend servers handling user requests
* App Tier = Backend application servers
* Data Tier = Database servers
* Cache Tier = Caching layer (Redis, Memcache)
* Messaging Tier = Message queues (Kafka, RabbitMQ)
* Management Tier = Infrastructure management and monitoring
* Downstream Applications (Circles) = External applications this app calls
* Infrastructure (Rectangles) = Databases, caches, and queues
* Thick lines (===) = Application-to-application calls
* Regular lines (---) = Infrastructure dependencies
* Colors indicate security zones and tiers

# Architecture Overview

## Internal Application Architecture

The ACDA application is structured using a multi-tier architecture pattern. Each tier has specific responsibilities and communicates with adjacent tiers through well-defined interfaces.

* Web Tier: Handles incoming HTTP/HTTPS requests, load balancing, and SSL termination
* App Tier: Contains business logic, API endpoints, and application processing
* Data Tier: Manages persistent data storage and database operations
* Cache Tier: Provides high-speed data caching to reduce database load
* Messaging Tier: Handles asynchronous messaging and event processing
* Management Tier: Manages monitoring, logging, and operational tools

## External Dependencies

The application interacts with various external systems:

* Downstream Applications: Other applications that this app calls directly
* Databases: Persistent data stores (MySQL, PostgreSQL, etc.)
* Caches: In-memory data stores (Redis, Memcache)
* Message Queues: Asynchronous messaging systems (Kafka, RabbitMQ)

## Security Considerations

* All traffic between tiers should be encrypted using TLS/SSL
* Database access should be restricted to App Tier servers only
* Management ports should not be accessible from external networks
* Implement micro-segmentation with strict firewall rules between tiers
* Monitor all cross-tier traffic for suspicious patterns
* Apply principle of least privilege for all inter-tier communications