BizFirstAi

**Edge** Stream

Real-time Event Diagnostics & Stream Observability

# Server Dashboard

- Servers
  - Servers
  - My Groups
- Communication
  - Chat
- Diagnostics
  - Event Stream
  - Activity Stream
- ANCP Nodes
- Pipeline Management

anit

Disconnect

## Servers

**bas** connected

Messages:

## Groups for bas

+ Create Group

### Random

Members: 0 | Created: 4/15/2026 | Last Activity: 12:48:53 AM

Join Group

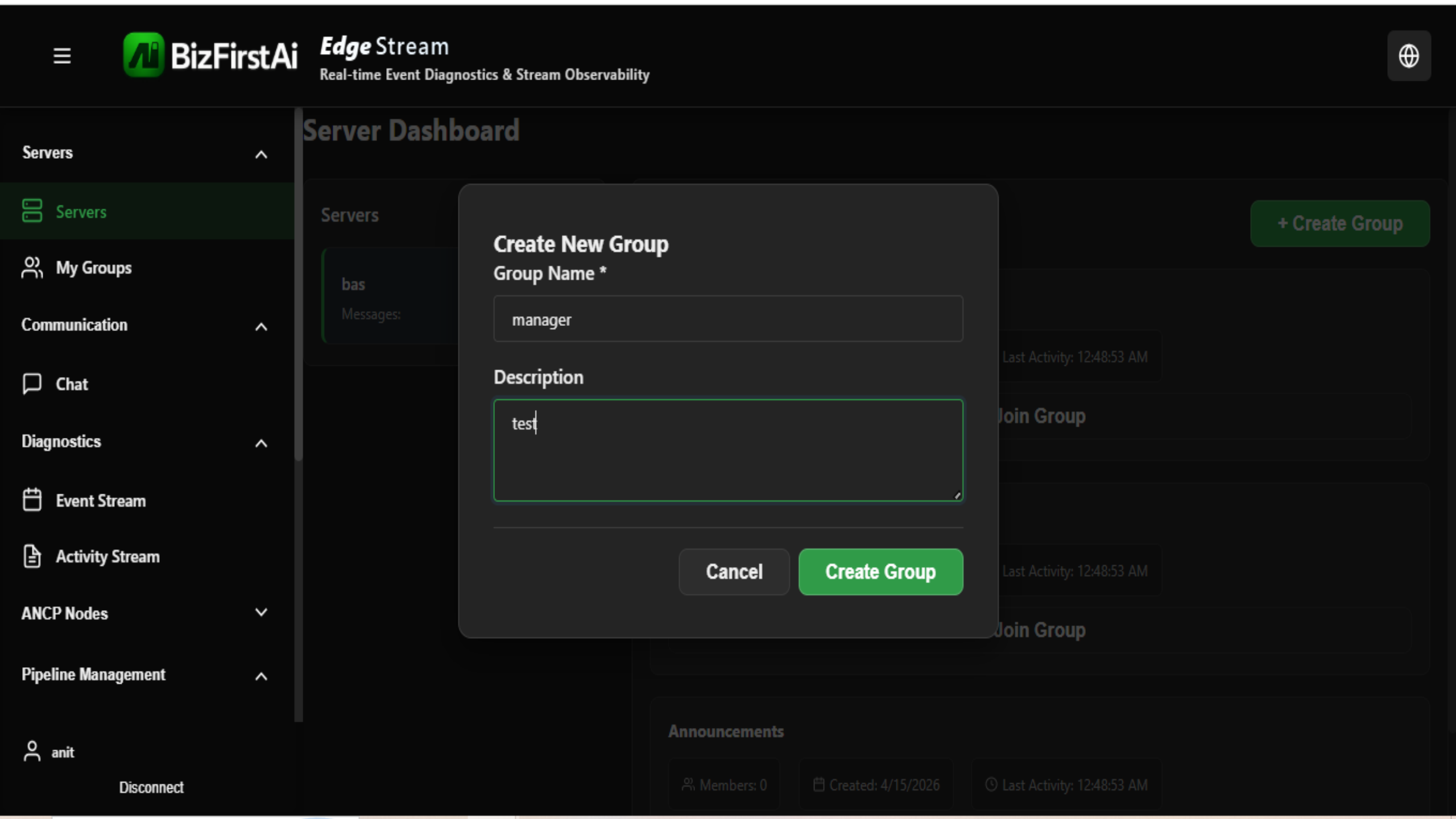
### General

Members: 0 | Created: 4/15/2026 | Last Activity: 12:48:53 AM

Join Group

### Announcements

Members: 0 | Created: 4/15/2026 | Last Activity: 12:48:53 AM


BizFirstAi
Edge Stream
Real-time Event Diagnostics & Stream Observability
Server Dashboard
Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Pipeline Management
anit
Disconnect
Servers
bas
Messages:
+ Create Group
Create New Group
Group Name *
manager
Description
test
Cancel
Create Group
Last Activity: 12:48:53 AM
Join Group
Last Activity: 12:48:53 AM
Join Group
Announcements
Members: 0
Created: 4/15/2026
Last Activity: 12:48:53 AM

Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Pipeline Management
anit
Disconnect

# My Groups

User ID: anit

## Active Groups (0)

You haven't joined any groups yet
Go to **Servers** dashboard to browse and join groups

Servers
Servers
My Groups
Communication
Chat
Diagnostics
ANCP Nodes
Pipeline Management
Pipelines
Available Hooks

anit
Disconnect

# Chat

SERVERS
1 server

bas
4 groups

No server selected

Select a server from the left panel to get started

Servers
Servers
My Groups
Communication
Chat
Diagnostics
ANCP Nodes
Pipeline Management
Pipelines
Available Hooks
anit
Disconnect

# Chat

SERVERS
1 server

bas
4 groups

## bas

ID: bas

GROUPS (4)

**Random**
0 members • Created by system

**General**
0 members • Created by system

**manager**
0 members • Created by anit
test

**Announcements**
0 members • Created by system

BizFirstAi

# Edge Stream

Real-time Event Diagnostics & Stream Observability

Servers

Servers

My Groups

Communication

Chat

Diagnostics

ANCP Nodes

Pipeline Management

Pipelines

Available Hooks

anit

Disconnect

SERVERS

1 server

bas
4 groups

Random

Stream

Random

Server: bas • User: anit • Connected (NOT in group)

0 messages

Log

Switch User

No messages yet. Start a conversation!

Logging enabled - check Inspector Logs page to view pipeline activity

Type a message...

Send

Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Pipeline Management

anit
Disconnect

# Event Stream

Monitor real-time event stream from EdgeStream

Filter by topic or server...

Total: 0 Filtered: 0

Clear Logs

BizFirstAi

**Edge** Stream

Real-time Event Diagnostics & Stream Observability

Servers

Servers

My Groups

Communication

Chat

Diagnostics

Event Stream

Activity Stream

ANCP Nodes

Pipeline Management

Pipelines

anit

Disconnect

# Activity Stream

Overview | Timeline | Details | Settings

| Total Messages | Successful | Failed | Avg Processing | Total Errors |
|---|---|---|---|---|
| 1 | 0 | 0 | 0ms | 0 |

| Topics |
|---|
| 1 |

**Slowest Message**

**0ms**

ID: ...

**Slowest Hook**

**0ms**

Hook:

Export Logs | Refresh | Clear All

BizFirstAi

**Edge** Stream

Real-time Event Diagnostics & Stream Observability

Servers

Servers

My Groups

Communication

Chat

Diagnostics

Event Stream

Activity Stream

ANCP Nodes

Pipeline Management

Pipelines

anit

Disconnect

# Activity Stream

Overview | Timeline | Details | Settings

Search by message ID, type, or topi

All Status

Filter by Server ID...

Filter by Topic...

ID msg-17761954...

processing

TYPE chat.message TOPIC group:bas:Random TIME ms 0 hooks 0 subs

Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Pipeline Management
Pipelines
anit
Disconnect

# Activity Stream

Overview | Timeline | Details | Settings

No event selected

Go to the **Timeline** tab and click an event to inspect its details.

Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Pipeline Management
Pipelines
anit
Disconnect

# Activity Stream

Overview
Timeline
Details
Settings

## Logging Configuration

- [x] Enable Hook Logging
  - [x] Log Message Body
    Includes full message content in logs
  - [x] Log Metadata
    Includes message metadata and attributes
- [ ] Enable Subscriber Logging
- [x] Enable Log Persistence

Max Log Entries: 1000

Servers
Servers
My Groups
Communication
Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
anit
Disconnect

## Panel A — Node Connection

Base URL: https://server2.bizfirst.interna

Node ID: 1

Authentication

- (•) None
- ( ) JWT
- ( ) ApiKey
- ( ) DID

Timeout (ms): 30000

Connect

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

## Pipeline Summary

| TOTAL PIPELINES | INCOMING | OUTGOING | BIDIRECTIONAL |
|---|---|---|---|
| 4 | 2 | 2 | 0 |

## Available Pipelines

Pipelines are routing and processing configurations that define how messages flow through the system.

All (4) | Incoming (2) | Outgoing (2) | Bidirectional (0)

All Sources (4) | EdgeStream (4)

incoming high EdgeStream

### Default Incoming Pipeline

`edgestream.default-incoming-v1`

Standard incoming pipeline: logging, validation, filtering, custom enrichment, and publish to subscribers.

incoming critical EdgeStream

### Security Incoming Pipeline

`edgestream.security-incoming-v1`

Security-first incoming pipeline. Applies strict validation before any enrichment. Recommended for untrusted sources.

outgoing high EdgeStream

### Default Outgoing Pipeline

`edgestream.default-outgoing-v1`

Standard outgoing pipeline: logging, validation, filtering, transformation (signing, classification, compression), and publish.

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

### Default Incoming Pipeline

edgestream.default-incoming-v1

Standard incoming pipeline: logging, validation, filtering, custom enrichment, and publish to subscribers.

v1.0.0 by EdgeStream standard incoming validation enrichment

Hooks: 5 · Subscribers: 1

### Security Incoming Pipeline

edgestream.security-incoming-v1

Security-first incoming pipeline. Applies strict validation before any enrichment. Recommended for untrusted sources.

v1.0.0 by EdgeStream security incoming strict-validation

Hooks: 4 · Subscribers: default

### Default Outgoing Pipeline

edgestream.default-outgoing-v1

Standard outgoing pipeline: logging, validation, filtering, transformation (signing, classification, compression), and publish.

v1.0.0 by EdgeStream standard outgoing transformation signing

Hooks: 5 · Subscribers: 1

outgoing medium EdgeStream

### Minimal Outgoing Pipeline

edgestream.minimal-outgoing-v1

Lightweight outgoing pipeline with only validation and publish. Suitable for high-throughput, low-latency scenarios where transformation overhead is undesired.

v1.0.0 by EdgeStream minimal outgoing performance lightweight

Hooks: 3 · Subscribers: default

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

# Available Hooks

Complete reference guide to all 14 available pipeline hooks. Hooks are composable pipeline stages that transform messages as they flow through the system.

## Pipeline Hooks Library

Each hook has a specific responsibility and execution priority. Hooks execute in order based on priority, and can abort or pause the pipeline as needed.

All (14) | Communication (1) | Diagnostic (1) | Integration (1) | Interactive (3) | Processing (1) | Resilience (1) | Security (4) | Terminal (1) | Validation (1)

Diagnostic

### LogHook

Logging and diagnostic purposes for debugging message flow

**Priority:** Medium

Processing

### FilterHook

Filter messages by predicate - aborts if predicate returns false

**Priority:** High

Validation

### ValidationHook

Validate message against JSON schema - aborts if validation fails

**Priority:** High

Security

### DecryptHook

Decrypt encrypted message body using

Security

### VerifySignatureHook

Verify message signature - aborts if signature

Communication

### AcknowledgeHook

Send acknowledgment back to server that

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

Security

### DecryptHook

Decrypt encrypted message body using configured cipher

**Priority:** High

Security

### VerifySignatureHook

Verify message signature - aborts if signature is invalid

**Priority:** High

Communication

### AcknowledgeHook

Send acknowledgment back to server that message was received

**Priority:** Medium

Interactive

### RenderToFormHook

Render form or dialog to user - can pause pipeline for user interaction

**Priority:** Low Can Pause Pipeline

Terminal

### PublishToSubscribersHook

Terminal hook - routes message to all matched topic subscribers

**Priority:** Medium

Security

### SignMessageHook

Sign outgoing messages with configured signature algorithm

**Priority:** Medium

Security

### EncryptMessageHook

Encrypt outgoing message body using configured cipher

**Priority:** Medium

Resilience

### RetryHook

Retry failed message sends with exponential backoff strategy

**Priority:** Low

Interactive

### UiQueryHook

Query UI component for user input and response handling

**Priority:** Low Can Pause Pipeline

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

Integration

### WebMcpHook

Integrate with WebMCP tools for extended functionality

**Priority:** Medium

Interactive

### FormResponseHook

Handle form submission responses and user input validation

**Priority:** Low

## Hook Execution Flow

Hooks execute in priority order, allowing early termination:

1 **Message Received**
Incoming envelope from server

↓

2 **Filter & Validate**
FilterHook, ValidationHook, VerifySignatureHook

↓

**Transform & Decrypt**

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

**2** **Filter & Validate**
FilterHook, ValidationHook, VerifySignatureHook

↓

**3** **Transform & Decrypt**
DecryptHook, LogHook, custom processing

↓

**4** **Interactive (Optional)**
RenderToFormHook, UiQueryHook - can pause pipeline

↓

**5** **Terminal Hook**
PublishToSubscribersHook - routes to subscribers

**Note:** If any hook returns `continue: false` or calls `abort()`, pipeline execution terminates at that point. Interactive hooks can pause execution, awaiting user input before resuming.

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

5 **Terminal Hook**
PublishToSubscribersHook - routes to subscribers

**Note:** If any hook returns `continue: false` or calls `abort()`, pipeline execution terminates at that point. Interactive hooks can pause execution, awaiting user input before resuming.

## Hook Categories

- **Communication** — 1 hook
- **Diagnostic** — 1 hook
- **Integration** — 1 hook
- **Interactive** — 3 hooks
- **Processing** — 1 hook
- **Resilience** — 1 hook
- **Security** — 4 hooks
- **Terminal** — 1 hook
- **Validation** — 1 hook

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

# Available Subscribers

Complete reference guide to message subscription patterns and subscriber API for the pub/sub system. Subscribers listen for messages on specific topics with support for wildcard matching.

## Subscription Patterns

The subscription system supports flexible topic matching with wildcards. Choose the pattern that best fits your message routing needs.

### *

Subscribe to all topics - matches any message

**Example:**

```
subscribe("*", callback)
```

### topic.subtopic

Subscribe to specific topic - exact match required

**Example:**

```
subscribe("form.submit", callback)
```

### topic.*

Subscribe to topic with wildcard - matches all subtopics

**Example:**

```
subscribe("form.*", callback)
```

### topic.sub.*

Subscribe to nested topics with wildcard matching

**Example:**

```
subscribe("data.user.*", callback)
```

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

## Subscription API

Use the SubscriptionManager to interact with the pub/sub system. Each server instance has its own subscription manager.

### Subscribe to Topic

Listen for messages on a specific topic pattern

```
const subscription = streamInstance
  .getServer('serverId')
  .subscriptionManager
  .subscribe('topic.name', (envelope) => {
    // Handle message
  })
```

### Publish to Topic

Send a message to all subscribers on a topic

```
await streamInstance
  .getServer('serverId')
  .subscriptionManager
  .publish('topic.name', envelope)
```

### Unsubscribe

Stop listening to messages on a topic

```
subscription.unsubscribe()
```

### Get Active Subscriptions

View all currently active subscriptions

```
const active = streamInstance
  .getServer('serverId')
  .subscriptionManager
  .subscriptions
```

## Message Delivery Flow

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

## Message Delivery Flow

Subscribers receive messages that match their topic patterns:

1. **Message Published**
   Message is sent to a topic via PublishToSubscribersHook

↓

2. **Pattern Matching**
   Topic is matched against all active subscriptions

↓

3. **Wildcard Resolution**
   Wildcard patterns are evaluated (*, topic.*, topic.sub.*)

↓

4. **Subscriber Notification**
   All matching subscribers receive the message

↓

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

**4** **Subscriber Notification**
All matching subscribers receive the message

↓

**5** **Callback Execution**
Subscriber callback function is invoked with message

**Note:** Multiple subscribers can listen to the same topic. A single message can be delivered to multiple subscribers if their patterns match. Subscribers are matched in registration order.

## Pattern Matching Examples

See how different patterns match against published topics:

**Pattern:** `form.*`

Matches:

- `form.submit` ✓
- `form.validate` ✓

**Pattern:** `data.user.*`

Matches:

- `data.user.created` ✓
- `data.user.updated` ✓

**Pattern:** `*`

Matches:

- `anything` ✓
- `any.topic` ✓

Chat
Diagnostics
Event Stream
Activity Stream
ANCP Nodes
Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
anit
Disconnect

Matches:

- `form.submit` ✓
- `form.validate` ✓
- `form.error` ✓
- `form.user.submit` ✗ (too deep)

Matches:

- `data.user.created` ✓
- `data.user.updated` ✓
- `data.user.deleted` ✓
- `data.user.profile.updated` ✗ (too deep)

Matches:

- `anything` ✓
- `any.topic` ✓
- `any.nested.topic` ✓
- Nothing is excluded ✓

## Best Practices

### ✓ Use Specific Patterns

Prefer specific topic patterns like `form.submit` over wildcards to reduce message overhead and improve performance.

### ✓ Unsubscribe When Done

Always unsubscribe from topics when they're no longer needed to free resources and prevent memory leaks.

### ✓ Handle Errors in Callbacks

Wrap subscriber callbacks in try-catch to prevent one subscriber from affecting others.

### ✓ Use Consistent Naming

Use hierarchical naming like `domain.action.detail` for topics to make patterns intuitive.

### ✓ Monitor Active Subscriptions

Use the monitoring tools to track active subscriptions and diagnose delivery issues.

### ✓ Test Wildcard Patterns

Verify your wildcard patterns match expected topics. Use the Subscribers page to monitor delivery.

Node Explorer
Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Demo
Help & Docs
anit
Disconnect

# Hook Execution Pipeline

Real-time monitoring of hook execution through the message processing pipeline

Live Monitor | Settings

| Total Hooks | Success | Failed |
|---|---|---|
| 0 | 0 | 0 |

Filter by hook name...

No hook executions recorded. Messages flow through hooks when received.

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

# Hooks Library - Reference Implementations

These hooks demonstrate the capability of the EdgeStream hook system. They are production-ready and can be configured and registered in your pipelines.

### LogHook

Diagnostic

Logs all messages flowing through the pipeline with optional filtering and payload capture.

### WebMcpHook

Integration

Generic REST API integration hook for calling external services, webhooks, and third-party APIs.

### SplunkLogHook

Observability

Specialized hook for streaming messages to Splunk HTTP Event Collector (HEC) for centralized logging.

## Global Observability Pattern (Recommended for Splunk)

Instead of hooking individual pipelines, implement observability at the stream level to capture ALL messages and hook executions:

```
// Global observability - capture ALL messages going to Splunk
// This is separate from pipeline hooks - runs at stream level

import { createEdgeStream } from 'edge-stream-js';
import { SplunkLogHook } from '@edge-stream/hooks';

const stream = createEdgeStream();

const splunkHook = new SplunkLogHook({
  hecUrl: process.env.SPLUNK_HEC_URL!
```

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

```
const stream = createEdgeStream();

const splunkHook = new SplunkLogHook({
  hecUrl: process.env.SPLUNK_HEC_URL!,
  hecToken: process.env.SPLUNK_HEC_TOKEN!,
});

// Subscribe to ALL messages (not just pipeline-specific)
stream.on('message:received', async (envelope) => {
  await splunkHook.execute({ envelope } as any);
});

// This captures observability from every hook execution,
// every message, every transformation - system-wide visibility
```

### Why Global Observability?

- **Complete Visibility**: Captures every message across all pipelines and hooks
- **System-Wide Metrics**: Track throughput, latency, errors from all sources
- **Single Integration**: Configure Splunk once instead of per-pipeline
- **Non-Intrusive**: Doesn't require modifying pipeline configurations
- **Flexible Filtering**: Filter at the stream level, not hook level

## Hook Architecture

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

### Why Global Observability?

- **Complete Visibility**: Captures every message across all pipelines and hooks
- **System-Wide Metrics**: Track throughput, latency, errors from all sources
- **Single Integration**: Configure Splunk once instead of per-pipeline
- **Non-Intrusive**: Doesn't require modifying pipeline configurations
- **Flexible Filtering**: Filter at the stream level, not hook level

## Hook Architecture

### Hook Lifecycle

1. Pipeline receives message
2. Creates context with envelope
3. Executes hooks by priority (low→high)
4. Each hook returns HookResult
5. Result controls pipeline flow

### Execution Order

| | |
|---|---|
| Priority 10 | LogHook |
| Priority 15 | SplunkLogHook |
| Priority 50 | WebMcpHook |

### Design Principles

- ✓ Single responsibility
- ✓ Non-blocking execution
- ✓ Graceful degradation
- ✓ Configurable behavior
- ✓ Error isolation

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

- Error isolation

## Security Best Practices

### Token Management

- Store HEC tokens in environment variables
- Never commit tokens to version control
- Use .env.local for development
- Rotate tokens regularly

### Network Security

- Always use HTTPS for remote endpoints
- Validate SSL certificates
- Set appropriate timeouts
- Implement rate limiting

### Data Privacy

- Control what data is logged
- Exclude sensitive PII by default
- Use transform functions for filtering
- Audit external service access

## Learn More

For detailed implementation guides, check out:

• Source code: packages/hooks/src/• Development History: packages/hooks/DevelopmentHistoryLog.md• Hooks vs Pipelines: See HooksPage.tsx for comparison

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

# Understanding Message Subscriptions

Learn how to subscribe to messages, implement handlers, and build real-time applications with EdgeStream

## What Are Subscriptions?

Subscriptions are the way your application listens for messages flowing through EdgeStream. Think of them as event listeners that wake up when messages matching your topic pattern arrive.

```
stream.subscribe('demo', 'user.created', (envelope) => {}
  console.log('New user:', envelope.data);
})
```

## Topic Pattern Matching

Subscriptions support flexible pattern matching for routing messages:

**Wildcard: ***

Subscribes to ALL messages

**Exact Match: topic.subtopic**

Listen to specific topic

**Wildcard Suffix: topic.***

Listen to all subtopics

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

Subscribes to ALL messages

```
subscribe('*', handler)
```

Listen to specific topic

```
subscribe('user.created', handler)
```

Listen to all subtopics

```
subscribe('user.*', handler)
```

**Nested Wildcard: topic.sub.***

Listen to nested patterns

```
subscribe('data.user.*', handler)
```

## Common Topic Hierarchies

These are standard topics you'll encounter when building EdgeStream applications:

| Category | Topics | Description |
| --- | --- | --- |
| Forms | `form.*` | Form events (open, submit, close, validate) |
| User Data | `data.user.*` | User create, update, delete events |

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

| Category | Topics | Description |
|---|---|---|
| Forms | `form.*` | Form events (open, submit, close, validate) |
| User Data | `data.user.*` | User create, update, delete events |
| Pipeline | `pipeline.*` | Pipeline execution start, complete, error |
| Messages | `message.*` | Message sent, received, routed |
| Errors | `error.*` | Exceptions, validation failures, timeouts |

## Best Practices

### Be Specific

Prefer specific topics over wildcards. This improves performance and makes your code maintainable.

### Always Unsubscribe

Unsubscribe when done to prevent memory leaks and improve performance.

### Handle Errors

Wrap callback logic in try-catch to prevent unhandled exceptions.

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

## Best Practices

### Be Specific

Prefer specific topics over wildcards. This improves performance and makes your code maintainable.

### Always Unsubscribe

Unsubscribe when done to prevent memory leaks and improve performance.

### Handle Errors

Wrap callback logic in try-catch to prevent unhandled exceptions.

### Avoid Heavy Operations

Don't perform expensive computations in callbacks. Offload to background tasks.

### Use Wildcards Sparingly

Wildcard subscriptions (*) catch all messages. Use sparingly for performance.

### Avoid Loops

Don't publish messages from within subscription callbacks for the same topic.

## API Reference

### Subscribe to Messages

```
const subscription = streamInstance.subscribe(
  'serverId',
```

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

# API Reference

### Subscribe to Messages

```
const subscription = streamInstance.subscribe(
  'serverId',
  'topic.pattern',
  (envelope) => {
    // Handle message
    console.log('Received:', envelope.data);
  }
);
```

### Unsubscribe

```
// Stop listening
subscription.unsubscribe();

// Check if still active
if (!subscription.active) {
  console.log('Unsubscribed successfully');
}
```

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

# Understanding Pipelines

Learn how pipelines route messages, compose hooks, and define message flow through EdgeStream

## What Are Pipelines?

Pipelines are configurations that define how messages flow through EdgeStream. They specify routing rules, hook execution, and subscriber configuration for specific topics or patterns.

```
Pipeline incoming message on topic → Execute hooks → Route to subscribers
```

## Pipeline Types

EdgeStream supports different pipeline directions for different use cases:

### ↓ Incoming

Processes messages received from external systems

### ↑ Outgoing

Processes messages being sent to external systems

### ↔ Bidirectional

Handles both incoming and outgoing messages

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

Processes messages received from external systems

`Server → Hooks → Subscribers`

Processes messages being sent to external systems

`App → Hooks → Server`

Handles both incoming and outgoing messages

`Both directions supported`

## Pipeline Components

### 1. Metadata

Describes the pipeline (name, ID, description, direction, impact level)

### 2. Hooks

Composable stages that process messages (logging, filtering, validation, transformation)

### 3. Subscribers

Handlers that listen for messages matching topic patterns

## Hook Execution Order

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

## Hook Execution Order

Hooks execute in priority order, allowing early termination of the pipeline:

1 **Message Received**
Incoming message from server

2 **Execute Hooks**
Hooks run in priority order (low → high)

3 **Route to Subscribers**
Send to matching topic subscribers

✓ **Complete**
Pipeline execution finished

## Impact Levels

Pipelines are classified by their impact on system operation:

**Critical**
System failure if pipeline fails

**High**
Major features affected

**Medium**
Some features degraded

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

## Impact Levels

Pipelines are classified by their impact on system operation:

**Critical**

System failure if pipeline fails

**High**

Major features affected

**Medium**

Some features degraded

**Low**

Minor or no impact

## Best Practices

**Keep Hooks Focused**

Each hook should have a single responsibility

**Order Matters**

Hooks execute in priority order - plan accordingly

**Handle Errors**

Implement error handling in hooks to prevent pipeline failure

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

## Best Practices

### Keep Hooks Focused
Each hook should have a single responsibility

### Order Matters
Hooks execute in priority order - plan accordingly

### Handle Errors
Implement error handling in hooks to prevent pipeline failure

### Avoid Blocking
Don't perform heavy operations in hooks - offload to workers

### Monitor Performance
Track hook execution time to identify bottlenecks

### Test Thoroughly
Test pipeline configurations with realistic data and loads

## Real-World Example
A typical user creation pipeline:

```
Message: user.created {id, email, name}
```

Pipeline Management
Pipelines
Available Hooks
Available Subscribers
Hook Execution
Subscriber Delivery
Demo & Learning
Help & Docs
Hooks Library
Subscribers
Pipelines
anit
Disconnect

### Avoid Blocking

Don't perform heavy operations in hooks - offload to workers

### Monitor Performance

Track hook execution time to identify bottlenecks

### Test Thoroughly

Test pipeline configurations with realistic data and loads

## Real-World Example

A typical user creation pipeline:

```
Message: user.created {id, email, name}


1. LogHook → Log to console/file
2. ValidationHook → Validate email format
3. EncryptionHook → Encrypt sensitive data
4. SignMessageHook → Sign message
5. PublishToSubscribersHook → Send to listeners


✓ User creation event processed
```