

# THE WIDGET WEEKLY

VOLUME 4, ISSUE 2 | FEBRUARY 2026

ACME

PLATFORM UPDATES

## Scaling the Widget API: A Move to V3

As the Acme Widget Platform continues its rapid growth, our engineering team is excited to announce the upcoming transition to **Widget API v3**. This major update addresses the scalability challenges of our legacy v2 architecture and introduces several highly requested features.

*“The move to v3 isn’t just an upgrade; it’s a fundamental reimagining of how widgets interact with global state.”*

The primary shift in v3 is the move from polling-based updates to a reactive, event-driven model. By leveraging our new IF-WS WebSocket layer, clients can now receive state changes in real-time with sub-100ms latency.

### Key Improvements in V3

- **Fine-grained Scopes:** Restrict API keys to specific widget types or even individual widget IDs.
- **Batch Processing:** Create up to 500 widgets in a single atomic request.
- **Improved Rate Limiting:** Dynamic limits based on current system load and subscription tier.

### IN THIS ISSUE

- **V3 Roadmap** ..... Page 1
- **WebSocket Best Practices** .. Page 2
- **New SDK for Rust** ..... Page 2
- **Developer Spotlight** .... Page 3
- **AcmeCon 2026 Dates** ..... Page 4

TUTORIAL

## Mastering Real-Time Sync

Connecting your dashboard to our WebSocket service (IF-WS) is now easier than ever. The new `awp-sync`

library handles reconnection, backoff, and state reconciliation automatically.

Architecture Diagram Placeholder

Figure 1: The new WebSocket sync flow

To get started, simply initialize the client:

```
const client = new AwpClient({
  apiKey: 'YOUR_KEY',
  realtime: true
});

client.on('widget.update', (evt) => {
  console.log(`Widget ${evt.id} changed!`);
});
```

COMMUNITY

## AcmeCon 2026

Mark your calendars! AcmeCon is returning to San Francisco this June. Join 5,000 other developers for three days of deep dives, networking, and the reveal of our new “Smart Widget” engine. Early bird tickets are available until March 15th.

### QUICK TIP

Did you know you can use `ETag` headers in V2 to avoid unnecessary data transfer? Always check for a `304 Not Modified` response to save bandwidth!