How Does Synchronization Work in New Exalate?

Last Modified on 02/03/2026 10:43 am EST

The replica-based architecture of Exalate

Exalate synchronization uses a replica-based architecture where each side independently controls what data to share and how to apply incoming information. When a work item changes in one system, the outgoing script on that side populates a replica object with the fields and values you've configured to share. This replica travels to the receiving system where the incoming script reads the data and applies it to the corresponding work item according to local rules.

How synchronization triggers and scripts work

The process begins when a trigger condition is met, such as a work item matching a specific project, label, or status filter. Exalate detects the change, executes the outgoing script to create the replica, and sends it to the connected system. The receiving side processes the incoming replica through its incoming script, which determines how to map fields, transform values, and update the local work item.

Independent control for outgoing and incoming scripts

Both outgoing and incoming scripts run independently on each side of the integration. This autonomous control means your organization decides what to send without needing permission from the receiving side, and they independently decide how to apply your data without your involvement. Neither side sees or modifies the other's scripts, maintaining separation and security especially important in cross-company integrations.

Real-time synchronization and conflict resolution

Synchronization happens in real time with changes propagating immediately after they occur. When updates happen on both sides simultaneously, Exalate processes them in sequence to maintain data consistency. The platform includes conflict resolution mechanisms you can configure in your scripts to handle cases where both sides modify the same field.

Helper functions for common tasks

Helper functions simplify common synchronization tasks like merging comments, handling attachments, and managing user mappings. The commentHelper.mergeComments function combines comment threads from both sides, attachmentHelper.mergeAttachments synchronizes files, and nodeHelper methods map entities like users, statuses, and priorities between different systems.

Queuing mechanism for network disruptions

If one system is temporarily unavailable, Exalate queues pending synchronization operations. When connectivity restores, queued updates process automatically to bring systems back into alignment without data loss. This queuing mechanism ensures reliable synchronization even when networks experience temporary disruptions or systems undergo maintenance.

Pricing and Licensing

Resources

Subscribe for a weekly Exalate hack

Academy

Blog

YouTube Channel

Ebooks

Still need help?

Join our Community

Visit our Service Desk

Find a Partner