Synchronization Overview

The ability to synchronize data between several Teamwork Servers is available as of version 17.0.3. One Teamwork Server can now expose and make projects from different servers accessible, along with owned projects. Server users can access remote projects from different locations in read-only mode and can perform any read-only actions.

There can be more than two servers in the pool. Each server can separately synchronize data from many other servers.

Usage Scenarios, Multisite Deployment

The main usage scenario for the inter-server synchronization capability is a multisite Teamwork Server deployment.

Very often a large company has several different sites, often in different countries, where projects are developed. Usually, the site has very good internal connectivity - LAN-class speeds (1Gbit/s, or at least multiMbit/s) are available on the site. However, connectivity between the sites is not as good - WAN-class speeds (several Mbit/s) are available between sites.

At times, developers at one site are responsible for the development of subsets of projects, while developers at other sites only need to use (but not modify) the artifacts produced by the first team.

In earlier versions, the company had to choose one site to deploy a single Teamwork Server and set up remote connections for developers joining the other sites.

The server performance (project open, commit, update, element locking times) for onsite developers were good. However, offsite developers suffered from degraded performance due to reduced connectivity parameters. This is especially important when managing large projects.

As of this version, the company can deploy many Teamwork Servers - one per site. Offsite developers can connect to their local server and work at LAN-speeds. Synchronization between servers can then be set up.

You can choose the synchronization frequency as necessary - hourly, daily, weekly

You can also set the specific time to perform the synchronization. For example, you may choose to synchronize at night, when internet traffic is reduced.

With synchronization set up, developers can access the projects of other teams on their local server. Any action that does not change the project is acceptable. Users can open the remote projects, browse, search, analyze then, generate reports, and, most importantly, integrate remote projects as used projects/libraries into their projects.

For example, if a team at one site is responsible for developing a requirements project, the team at another site can take this requirements project and incorporate it into their implementation project(s)

Please note that if developers at different sites edit the same project, one server will be designated as the home server of the project. Any developer who needs to edit the project must log onto the home server of the project to edit it.

There are additional, less frequent scenarios when synchronization between the servers can be handy. For example, if a small number of developers generate a project set that must be readable by a large number of users, the company may want to set up a small main development server where developers will not be affected by network traffic, as well as a large server for handling all read-only traffic (possibly in a different network security zone, etc.).

The licensing scheme is straightforward. Each Teamwork Server deployment is licensed separately. Synchronizing multiple sites is NOT counted as one big deployment, but as many separate deployments. There are no additional charges (such as a separate "enterprise" edition of the server) for using the synchronization feature.

Characteristics

The synchronization process characteristics are:

- Synchronized items are Projects and Categories. The entire project history is synchronized with all commit data, including all branches and comments. The user's data is NOT synchronized. Each server has its independent user list, and the administrator of each server assigns permissions to each server.
- Each project has Server-Affinity, with one writer and (potentially) many readers. Each project has a home server. Projects can be edited only on the home server. All other servers can only read this project (open the project, use it as a read-only used project/ library of the other projects and so on). Having a single writer prevents incompatible modification conflicts between the servers without introducing any additional burden, such as inter-server locks.
- Synchronization is Incremental. The synchronization process can be run many times. Each synchronization process updates the target server to the latest data from the source server. The amount of the data transferred is proportional to any changes since the last synchronization.
- Synchronization is Asynchronous. Servers continue to run asynchronously. Changes on one server are propagated to other servers only during the synchronization procedure. The synchronization procedure can be arbitrarily delayed from the moment changes occur (for example only monthly synchronization).
- Synchronization is Batched; it is not a continuous process. It has clearly defined start and stop points. However, the frequency of synchronization can be arbitrary (for example every 15 minutes).
- Synchronization is Non-Intrusive. While synchronization is running, users can continue to work on both the source and the target servers. It is not necessary to restart the target server after finishing synchronization.

> **Looging out**
>
> Make sure all users are logged out before restarting the server.

- Synchronization is Separate in each Direction. Synchronization between each source and target server pair is independent of any different server pair. This includes synchronization in the opposite direction. If server S1 is synchronized from server S2, this does not automatically mean that S2 is synchronized from S1. Synchronization in the opposite direction can be set up on a different schedule or disabled.
- Target server is Active. The target server (the one receiving data) drives the synchronization process. It connects to the source server, determines what updates need to be transferred, fetches the data and incorporates it into its repository.
- Synchronization is Direct only. Currently, the synchronization procedure can only take projects directly from their home server. Transferring projects through an intermediate server chain (server S1'server S2'server S3) is not supported.
- Synchronization can be Partial. The administrators can specify that all the projects of the source server should be synchronized to the target server OR specify a subset of projects to synchronize.