Best integration practices

Article • 11/10/2023

Azure DevOps Services

Tools and integrations between services improve efficiencies for Azure DevOps Services. If you aren't careful, automated tools can get out of control executing a high rate of requests. The requests cause Azure DevOps to enforce rate limits to your organization. To help reduce your risk of hitting the rate limits, follow these best practices when you're using the REST APIs to integrate with Azure DevOps.

Push only actionable work items

Only push actionable items into Azure DevOps that your team plans to engage on or address in the future. Keep work items out of Azure DevOps until necessary. For example, don't attempt to store telemetry data in Azure DevOps.

Maintain your own data store

Don't add work items into Azure DevOps for the sake of having them all in one place. Azure DevOps Services isn't designed as a data storage service. Maintain your own data store.

Batch your changes

Doing single operations is slow and expensive, which is the leading cause for performance issues and rate limiting. Batch your changes into a single call. For more information, see our batch documentation and sample code.

Limit your revisions

Many revisions on a single work item create bloat and cause performance problems. We recommend doing the following tasks:

- Reduce your updates by batching your field changes. Don't update just one field at a time.
- If you have changes to multiple work items, batch those changes into a single action.
- Keep the number of revisions to a minimum to avoid revision limits.

> **Note**
>
> A work item revision limit of 10,000 is in effect for updates made through the REST API. This limit restricts updates from the REST API, but, updates from the web portal aren't affected.

Optimize queries

Optimize your queries to return a modest number of results. Complex conditions and filters can lead to long-running queries. Keep your queries execution time under 30 seconds to avoid threshold failures.

Query performance tips

- Place a date or range-limiting clause near the top of a query whenever possible.
- Reduce the number of clauses that use the *Ever* operator.
- Reduce the number of clauses that use the *Contains* operator, except for Tags.
  - Use the *Contains Words* operator when available.
  - Don't use the *Contains* operator on long text fields, as it's expensive.
- Avoid the '<>' and not operators when possible.
- Avoid using the *In Group* operator for large groups.
- Minimize the number of Or operators and ensure you still have top-level scoping before using.
- Avoid using an *OR* clause between an *In Group* operator and Area or Iteration Paths.
- Reduce the number of overall clauses to achieve your goal when possible.
- Avoid sorting on anything other than core fields, such as *ID*, when possible.
- Use a custom field in your filters if you want to sort on a custom field.
- Specify a project if possible. Otherwise, the query gets scoped to the entire collection and could take dramatically longer than it needs to. Uncheck "Query across projects on the top-right corner" of the query editor.

Query across projects

- Specify which project you're looking for if the query requires search across projects.
- Use *tags* instead of *keywords* when possible, unless you're searching for partial text of a string.

Handle failures gracefully

Updates and queries fail when resource limits or frequency of utilization crosses the limit threshold. For example, a query that runs longer than 30 seconds returns the following error:

```
VS402335: The timeout period (30 seconds) elapsed prior to completion of the query or the server is not responding.
```

When you're consuming the REST APIs, make sure you design your code to handle failures appropriately.

Limit your links

Limit the number of links per work item as much as possible, to avoid enforcement of link limits.

> **Important**
>
> We plan to enforce work item revision and link limits in the near future. These limits get determined by performance monitoring and customer feedback.

Don't use queries for reporting

Using queries and individual *get work item* calls is the top way to get rate limits enforced on your organization. Don't execute queries to return large lists of work items. Use the reporting work item links and work item revisions REST APIs instead.

For more information, see our C# sample on GitHub .

Related articles

- .NET client libraries
- Migration guide
- Service hooks
- REST API versioning