# Updating Maintenance window is stuck

Last updated by | Luis Humberto Aranda Cardenas | Nov 11, 2022 at 9:03 AM PST

Contents



## Issue

Customers may notice that updating the Maintenance window on their Azure SQL DB/Managed Instance is taking forever or appears to be stuck.

## Investigation/Analysis

The estimated time taken for this operation is similar to a scaling operation. The estimated timelines for Azure SQL DB operations is listed here: Scaling Azure SQL DB and the timeline for Azure SQL Managed Instance can be found here: Overview of Azure SQL Managed Instance management operations.

If updating the Maintenance window operation is ongoing and has not hit these timelines, customers will have to have for the operation to finish. If the operation has passed these estimated timelines, check if the operation is stuck or has hit an internal failure.

First, verify the stuck workflow in Kusto via ASC under Provisioning tab. The operation can be seen with a startTime, but no end time. Below MonManagement Kusto query can also help verify the ongoing operation:

```
let starttime = {StartTime};
            let endtime = {EndTime};
            let subscription = '{subscriptionid}';
            let DBorMI = '>{serverorDBName}<';
            MonManagementOperations
            | where originalEventTimestamp >= starttime and originalEventTimestamp <= endtime
            | where subscription_id =~ subscription or SubscriptionId =~ subscription
            | where event == 'management_operation_start'
            | where operation_parameters has '<MaintenancePolicyId>'
            | where operation_parameters contains DBorMI
            | project originalEventTimestamp, request_id, operation_type, operation_parameters
            | join kind = leftouter
            (
            MonManagementOperations
            | where originalEventTimestamp >= starttime
            and originalEventTimestamp <= endtime //remove to check for operations completing after EndTime
            | where subscription_id =~ subscription or SubscriptionId =~ subscription
            | where event != 'management_operation_start'
            | where operation_parameters has '<MaintenancePolicyId>'
            | where operation_parameters contains DBorMI
            | project originalEventTimestamp, request_id, event,error_code, error_message, operation_result
            ) on request_id
            | extend Result = replace(@'management_operation_(.*)', @'\1', event)
            | extend Input = tostring(parse_xml(operation_parameters))
            | extend InstanceOperationType = parse_json(Input).InputParameters.OperationType
            | extend MaintenancePolicyId = tostring(parse_json(Input).InputParameters.MaintenancePolicyId)
            | project originalEventTimestamp, originalEventTimestamp1, request_id, InstanceOperationType, Main
            | project-rename StartTime = originalEventTimestamp, EndTime = originalEventTimestamp1
            | order by StartTime asc
            | join kind=leftouter (
            MonManagement
            | where TIMESTAMP >= starttime and TIMESTAMP <= endtime
            | where event  == 'maintenance_window_info'
            | where isnotempty(maintenance_policy_id)
            | summarize arg_max(TIMESTAMP,scheduler_pattern) by maintenance_policy_id
            | extend SchedulerPattern = parse_xml(scheduler_pattern)
            | extend ScheduledDays = tostring(SchedulerPattern.WindowScheduler.ScheduledDays.DayOfWeek)
            | extend Frequency = SchedulerPattern.WindowScheduler.Frequency
            | extend TimeZone = SchedulerPattern.WindowScheduler.TimeZone
            | extend Time = SchedulerPattern.WindowScheduler.Time
            | extend Duration = SchedulerPattern.WindowScheduler.Duration
            | project maintenance_policy_id, ScheduledDays, Frequency, Time, Duration
            ) on $left.MaintenancePolicyId == $right.maintenance_policy_id
            | project StartTime, EndTime, request_id, InstanceOperationType, Result, MaintenancePolicyId, Sche
            | order by StartTime asc
```

Once it is confirmed that the operation is ongoing, verify if the operation is stuck due to 'Lock' on resources:

```
let customerSubscriptionId = "{subscriptionid}";
let resourceGroupName1 = "{ServerDBResourceGroupName}";
let resourceGroupName2 = "{vnetResourceGroup}";       //If this is a Managed Instance and if the MI RG is diff
let starttime = {StartTime};
let endtime = {EndTime};
MonManagement
| where TIMESTAMP >= starttime and TIMESTAMP <= endtime
| where nrp_response_body has customerSubscriptionId
| where nrp_response_body has resourceGroupName1 or nrp_response_body has resourceGroupName2
| where nrp_response_body has 'lock'
| where event == "nrp_failed"
| project TIMESTAMP, nrp_response_body, message, operation, level, request_url, response_status, network_resou
```

Sample output:

| TIMESTAMP | nrp_response_body |
| --- | --- |
| 2022-03-31 10:35:37.4653614 | {"error":{"code":"ScopeLocked","message":"The scope '/subscriptions/{subscriptionid}/resourceGroups/cash-nonprod-host/providers/Microsoft.Network/serviceEndpointPolicies/_e41f87a2_pdc_ea034c7a23-4891-99f9-2cbfd79df307' cannot perform delete operation because followin scope(s) are locked: '/subscriptions/{subscriptionid}/resourceGroups/cash-nonproc host'. Please remove the lock and try again."}} |

The nrp_response_body shows that there is a lock on a specific resource that is preventing the Maintenance operation from proceeding.

## Mitigation

Customer need to remove the lock on the specified resource for the Maintenance operation to proceed and finish.

**How good have you found this content?**