Import or Export Array Dimensions exceeded support range

Last updated by | Keith Elmore | Apr 5, 2021 at 7:56 AM PDT

Contents

Issue

Importing database via the portal fails with an error message as follows

"Could not export schema and data from database. One or more errors occurred. One or more errors occurred. One or more errors occurred. One or more errors occurred. One or more errors occurred. Array dimensions exceeded supported range."

Exporting from sqlpackage.exe can also fail with the following

Microsoft.Data.Tools.Diagnostics.Tracer Error: 19 : 2021-01-30T02:56:55 : Retry requested: Retry count = 1. Delay = 00:00:00, SQL Error Code = , SQL Error Number = , Can retry error = False, Will retry = False
System.OutOfMemoryException: Array dimensions exceeded supported range.
at System.Data.SqlClient.TdsParser.TryReadPlpUnicodeChars(Char[]& buff, Int32 offst, Int32 len, TdsParserStateObject stateObj, Int32& totalCharsRead)
at System.Data.SqlClient.TdsParser.TryReadSqlStringValue(SqlBuffer value, Byte type, Int32 length, Encoding encoding, Boolean isPlp, TdsParserStateObject stateObj)
at System.Data.SqlClient.TdsParser.TryReadSqlValue(SqlBuffer value, SqlMetaDataPriv md, Int32 length, TdsParserStateObject stateObj, SqlCommandColumnEncryptionSetting columnEncryptionOverride, String columnName)
at System.Data.SqlClient.SqlDataReader.TryReadColumnInternal(Int32 i, Boolean readHeaderOnly)
at System.Data.SqlClient.SqlDataReader.TryReadColumn(Int32 i, Boolean setTimeout, Boolean allowPartiallyReadColumn)
at System.Data.SqlClient.SqlDataReader.GetSqlString(Int32 i)
at System.Data.SqlClient.SqlDataReader.GetChars(Int32 i, Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)
at Microsoft.Data.Tools.Schema.Sql.SqlClient.Bcp.TextTypesSerializer.<GetByteChunks>d__d.MoveNext()
at Microsoft.Data.Tools.Schema.Sql.SqlClient.Bcp.VarCharSerializer.<GetBytes>d__0.MoveNext()
at Microsoft.Data.Tools.Schema.Sql.SqlClient.Bcp.SqlRawStream.Read(Byte[] buffer, Int32 offset, Int32 count)
at System.IO.Stream.InternalCopyTo(Stream destination, Int32 bufferSize)
at Microsoft.Data.Tools.Schema.Sql.Dac.Data.Export.ExportTableBatchHelper.WriteRow(DataReaderIterator dataReaderIterator)
at Microsoft.Data.Tools.Schema.Sql.Dac.Data.Export.ExportTablePartHelper.ProcessDataBatch(DataReaderIterator

dataReaderIterator, RetryState retryState)
at Microsoft.Data.Tools.Schema.Sql.Dac.Data.Export.ExportTablePartHelper.ExportTableWithRetryState(RetryState retryState)
at Microsoft.Data.Tools.Schema.Common.SqlClient.RetryPolicy.<>c__DisplayClass4.<ExecuteAction>b__3(RetryState retryState)
at Microsoft.Data.Tools.Schema.Common.SqlClient.RetryPolicy.ExecuteAction[R](Func `2 func, Nullable` 1 token)
Microsoft.Data.Tools.Diagnostics.Tracer Error: 19 : 2021-01-30T02:56:55 :

Investigation/Analysis

The following query can be used to find the failed imports

```
MonManagementOperations
| whereTIMESTAMP >= ago(10d)
| whereoperation_parameterscontains'>c7503184-1cfb-49a3-8452-a6324565c1b8<'andoperation_parameterscontains'>abco01mstr1c52wprod<'andoperation_type == 'ExportDatabase'
| extendd=parse_xml(operation_parameters)
|
extendServerName=iif(isempty(tostring(d.InputParameters.LogicalServerName)),tostring(d.InputParameters.ServerName),tostring(d.InputParameters.LogicalServerName))
| extendCreateDatabaseName = tostring(d.InputParameters.DatabaseName)
| extendLogicalDatabaseName = tostring(d.InputParameters.LogicalDatabaseName)
| extendServiceLevelObjective=tostring(d.InputParameters.ServiceObjectiveName)
| extendDatabaseName = iif(isempty(CreateDatabaseName), LogicalDatabaseName, CreateDatabaseName)
| projectTIMESTAMP, request_id, message, event, operation_type, ServerName, DatabaseName, ServiceLevelObjective
| joinkind=inner (
MonManagement
| summarizemin(originalEventTimestamp), max(originalEventTimestamp), max(service_level_objective_name), max(serviceLevelObjective) byrequest_id
) onrequest_id
| joinkind=inner(
MonManagementOperations
| summarizemake_list(event), make_list(PreciseTimeStamp), min(PreciseTimeStamp), arg_min(operation_parameters, PreciseTimeStamp), any(operation_category) byrequest_id, operation_type
| orderbymin_PreciseTimeStampdesc
| extendongoing_operation = array_length(list_event) == 1 andlist_event[0] == 'management_operation_start'
| extendsucceeded_operation = array_length(list_event) >= 1 andlist_eventcontains'management_operation_success'
| extendoperation_timed_out = array_length(list_event) >= 1 andlist_eventcontains'management_operation_timeout'
| extendongoing_operation = case(ongoing_operation == 0, 'False', 'True')
| extendsucceeded_operation = case(succeeded_operation == 0, 'False', 'True')
| extendoperation_timed_out = case(operation_timed_out == 0, 'False', 'True')
| extendevent_list = tostring(list_event)
| extendevent_timestamp = tostring(list_PreciseTimeStamp)
| project-renameTimeStamp = min_PreciseTimeStamp
| project-renameoperation_category = any_operation_category
```

```
| project-awaylist_event, list_PreciseTimeStamp, PreciseTimeStamp, event_timestamp
| projectTimeStamp, operation_type, operation_parameters, request_id, ongoing_operation,
succeeded_operation, operation_timed_out
) onrequest_id
| extendelapsed_time_min=datetime_diff('minute', max_originalEventTimestamp, min_originalEventTimestamp)
| summarizebystart_time=min_originalEventTimestamp, last_event_time=max_originalEventTimestamp
, elapsed_time_min, request_id, operation_type, ServerName, DatabaseName
, succeeded_operation, ServiceLevelObjective
```

Mitigation

There is a workaround to use the x64 version of SqlPackage:

https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/file-schema/runtime/gcallowverylargeobjects-element

1. Open the sqlpackage.exe.config file in notepad located at "C:\Program Files\Microsoft SQL Server\150\DAC\bin\SqlPackage.exe.config"
2. You will need launch notepad as an elevated administrator process to edit this file
3. Add the gcAllowVeryLargeObjects XML element under the configuration/runtime element: Save the file

```
<configuration>
        <runtime>
            <gcAllowVeryLargeObjects enabled="true" />
```

Once this is implemented the import should complete without this error.

Root Cause Classification

Cases resolved by this TSG should be coded to the following root cause:

Root Cause: Azure SQL v3\Import/Export\Import issues\Import failed

How good have you found this content?