**A quick and dirty scan of a list of instances using a dynamic linked server.**

[Kenneth Fisher](https://www.sqlservercentral.com/author/kennethfisher), 2020-11-23 (first published: 2020-11-12)

FROM: <https://www.sqlservercentral.com/blogs/a-quick-and-dirty-scan-of-a-list-of-instances-using-a-dynamic-linked-server>

Note: This is not exactly a *dyanmic* linked server. It just gets dropped and recreated in a loop.

I recently did a post on [testing a linked server](https://sqlstudies.com/2020/10/27/how-do-i-test-if-a-linked-server-works-using-t-sql/) where I said I would explain *why* I wanted to make the test. Basically I needed to scan a few hundred instance names and do the following

* Check if the instance is one we have access to or even exists? If not make a note of the error so we can tell the difference.
* Collect information like instance size (total size of all databases), CPU count, memory count etc.
* Collect a list of database names on the instance, their status, size, etc.

So the first thing I did was throw the list of instance names into a cursor and then put the code from my last post inside the loop.

-- Create MyLinkedServer using the current server so that it exists and

-- the code will compile.

EXEC master.dbo.sp\_addlinkedserver @server = N'MyLinkedServer', @srvproduct=N'', @provider=N'SQLNCLI', @Datasrc = @@SERVERNAME;

GO

-- Create temp table to hold instance names.

-- You'll probably want a permanent table.

CREATE TABLE #InstanceList (InstanceName NVARCHAR(256));

INSERT INTO #InstanceList VALUES

('InstanceA'),

('InstanceB'),

('InstanceC');

-- Declare vars

DECLARE @sql NVARCHAR(max);

DECLARE @InstanceName NVARCHAR(256);

-- setup cursor to loop through servers

DECLARE InstList CURSOR FOR

SELECT InstanceName

FROM #InstanceList;

OPEN InstList

FETCH NEXT FROM InstList INTO @InstanceName

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN TRY

EXEC master.dbo.sp\_addlinkedserver @server = N'MyLinkedServer', @srvproduct=N'', @provider=N'SQLNCLI', @Datasrc = @InstanceName;

-- Test the linked server.

EXEC sp\_testlinkedserver @server = N'MyLinkedServer'

EXEC master.dbo.sp\_addlinkedsrvlogin @rmtsrvname=N'MyLinkedServer',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL

END TRY

BEGIN CATCH

INSERT INTO dbo.[LinkedServerLog] VALUES (

@InstanceName

,ERROR\_NUMBER()

,ERROR\_SEVERITY()

,ERROR\_STATE()

,ERROR\_PROCEDURE()

,ERROR\_LINE()

,ERROR\_MESSAGE());

END CATCH

FETCH NEXT FROM InstList into @InstanceName;

END

CLOSE InstList;

DEALLOCATE InstList;

-- Cleanup

IF EXISTS (SELECT \* FROM sys.servers WHERE name = 'MyLinkedServer')

EXEC master.dbo.sp\_dropserver @server=N'MyLinkedServer', @droplogins='droplogins';

I now have a piece of code that loops through a list of instances and creates a linked server for each one. It then tests that linked server to make sure I can connect and if I can’t store the error into an error table. From there I can see which instances I couldn’t connect to and which I could connect but couldn’t log into.

Now, all I have to do is add code into the try block that uses the linked server to collect information.

-- Create MyLinkedServer using the current server so that it exists and

-- the code will compile.

EXEC master.dbo.sp\_addlinkedserver @server = N'MyLinkedServer', @srvproduct=N'', @provider=N'SQLNCLI', @Datasrc = @@SERVERNAME;

GO

-- Create temp table to hold instance names.

-- You'll probably want a permanent table.

CREATE TABLE #InstanceList (InstanceName NVARCHAR(256));

INSERT INTO #InstanceList VALUES

('InstanceA'),

('InstanceB'),

('InstanceC');

-- Create temp table to hold database sizes.

CREATE TABLE #DBList (

InstanceName NVARCHAR(256),

DatabaseName NVARCHAR(256),

DatabaseSize DECIMAL(17,5)

);

-- Declare vars

DECLARE @sql NVARCHAR(max);

DECLARE @InstanceName NVARCHAR(256);

-- setup cursor to loop through servers

DECLARE InstList CURSOR FOR

SELECT InstanceName

FROM #InstanceList;

OPEN InstList

FETCH NEXT FROM InstList INTO @InstanceName

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN TRY

IF EXISTS (SELECT \* FROM sys.servers WHERE name = 'MyLinkedServer')

EXEC master.dbo.sp\_dropserver @server=N'MyLinkedServer', @droplogins='droplogins'

EXEC master.dbo.sp\_addlinkedserver @server = N'MyLinkedServer', @srvproduct=N'', @provider=N'SQLNCLI', @Datasrc = @InstanceName;

-- Test the linked server.

EXEC sp\_testlinkedserver @server = N'MyLinkedServer'

EXEC master.dbo.sp\_addlinkedsrvlogin @rmtsrvname=N'MyLinkedServer',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL

INSERT INTO #DBList

SELECT @InstanceName, dbs.name, SUM(size)/128/1024.0

FROM MyLinkedServer.master.sys.databases dbs

JOIN MyLinkedServer.master.sys.master\_files dbfiles

ON dbs.database\_id = dbfiles.database\_id

GROUP BY dbs.Name, dbs.database\_id;

END TRY

BEGIN CATCH

INSERT INTO dbo.[LinkedServerLog] VALUES (

@InstanceName

,ERROR\_NUMBER()

,ERROR\_SEVERITY()

,ERROR\_STATE()

,ERROR\_PROCEDURE()

,ERROR\_LINE()

,ERROR\_MESSAGE());

END CATCH

FETCH NEXT FROM InstList into @InstanceName;

END

CLOSE InstList;

DEALLOCATE InstList;

-- Cleanup

IF EXISTS (SELECT \* FROM sys.servers WHERE name = 'MyLinkedServer')

EXEC master.dbo.sp\_dropserver @server=N'MyLinkedServer', @droplogins='droplogins' ;

A couple of notes here. I have a piece of code at the top that adds *MyLinkedServer* to make sure it exists when the code starts. Otherwise it won’t compile. Also, because of the connection time for the test, particularly if you have a bunch of instances that you can’t log into/don’t exist, this script is going to take a while just to handle the loop. Make sure that the data collection code is hitting system tables and/or is as quick as you can make it.

Like I said in the title, this is pretty quick and dirty. This is the kind of thing you throw together because your manager wants some data collected from a bunch of sources ASAP and T-SQL is by far your best language. There are a lot of better ways to handle this.