OLAPRGateway

Connecting R to Microsoft data sources

Version 1.0.0.0 | August 6, 2015

License

General Description

OLAPRGateway enables the R Programing Language to communicate with native Microsoft technologies including SQL Server, SQL Azure, SQL Server Analytics Services/OLAP and custom .NET libraries. Using OLAPRGateway the R programmer can execute SQL and MDX queries and receive back the result set in the R language data.frame format. In addition to SQL related queries, OLAPRGateway also supports execution of native .NET code for R programmers though plug-ins .NET developers can create using the supplied OLAPRGatewayExt base class.Architecture

Architecture

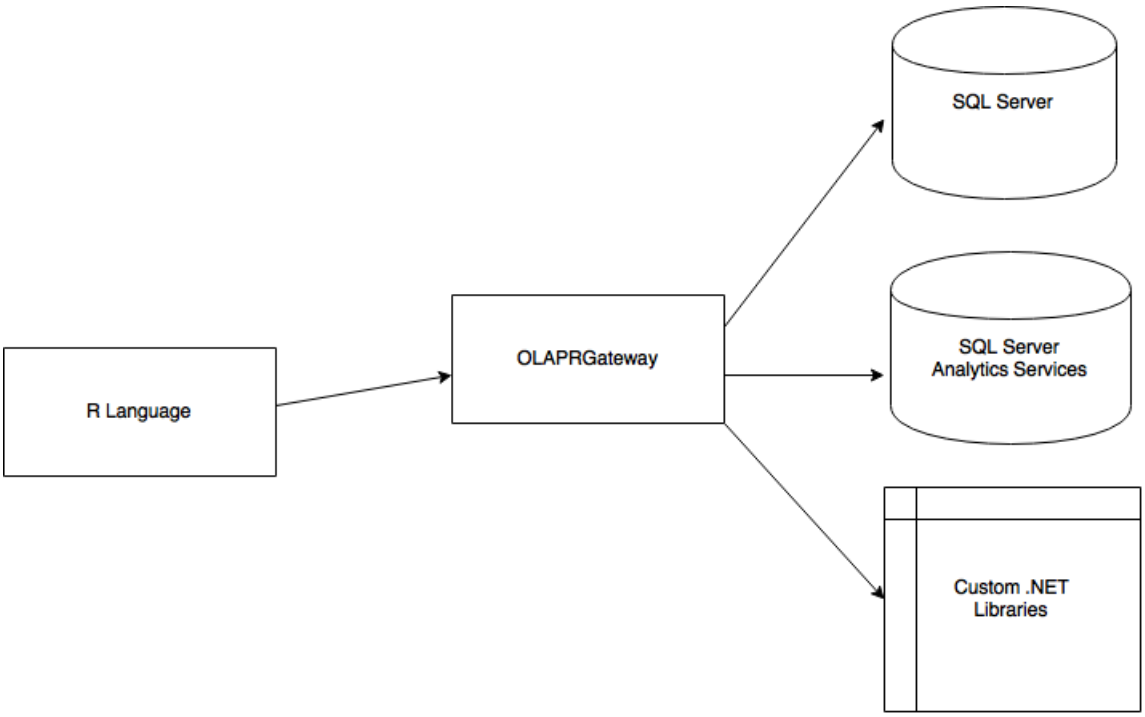
OLAPRGateway consist of a Windows console application and an R package. The R package provides R programmers with a collection of methods that can be called to query SQL Server, Analytics Services or to pass parameters to a custom .NET library and receive back a result set encapsulated in an R compatible data.frame object. To start the gateway the OLARRGateway Windows console application is launched. The OLAPRGateway console application initiates a TCP/IP listener on the configured port (8888 by default). The R language methods provided in the OLAPRGateway R package communicate with the OLAPRGateway console application. As part of this work flow the security token configured by the gateway is supplied with each R method call to the gateway. The console application submits SQL

and MDX queries to the desired servers and transforms the result set into an R language script needed to create a data.frame object.

In most scenarios the OLAPRGateway console application will run on the same system where the R language development is occurring. However this is not required. If the R programmer is using Mac OS they can choose to run the OLAPRGateway on a Windows that's also running on the network or cloud.

R Programmers who are committed to Mac OS and/or Linux also have the option to run Windows in a VM environment that supports virtual networking. For example using Oracle's VirtualBox software, a Mac OS user can install a Windows VM and install OLAPRGateway on the VM. Through virtual networking the R language OLAPRGateway methods running on the non-Windows OS can be made to communicate with the OLAPRGateway console running on the VM. In this way OLAP, SQL data and native .NET code can be executed though he R language even while running on non-Windows platforms.

Additional information on Oracle VirtualBox and configuring virtual networking for VirtualBox can be found here https://www.virtualbox.com and here https://www.virtualbox.org/manual/ch06.html



Configuration

OLAPRGateway.exe must run on a Windows machine that is able to connect to the target SQL Server or SSAServer (SQL Server Analytics Services). OLAPRGateway does not allow concurrent connections. It is suggested that you run OLAPRGateway.exe on the same machine running the R programming environment. If OLAPRGateway.exe must run on a different machine than where the R development is being performed (e.g. if your R programmers are on Mac OS X) then you must make sure local and LAN level Firewall rules allow traffic for the port you decide to use.

Configuration data is contained in the file **OLAPRGateway.exe.config**. The table below provides an overview of configuration options

Configuration key name	Description
SQLServer	Should contain the full ADO.NET connection string for the SQL Server or SQL Azure database. More information on SQL Connection strings can be found here https://msdn.microsoft.com/enus/library/jj653752(v=vs.110).aspx#sqldatabase
SSAServer	Should contain the full ADO.NET connection string for the SSAS Server. More information on SQL Connection strings for SSAS can be found here: https://msdn.microsoft.com/en-us/library/ms123468.aspx
Mode	Should be set to **SQLServer** or **SSAS** determines which connection string will be loaded and which processing method will be used. When SQLServer mode is specified all commands are processed as SQL and directed to the specified SQLServer. When SSAS mode is specified all commands are processed as MDX and sent to the specified SSAServer
Token	Used to authenticate request from the OLAPRGateway R library methods. The OLAPRGateway uses TCP/IP to communicate with it's R library counterpart. The security token helps insure the incoming request is from a valid R method call. It's suggested that you change the default Token value to something unique to your environment.
Port	The TCP/IP port the Gateway will run on. The default port is 8888. The OLAPRGateway R language methods will default to use port 8888. If you change the port, you must inform the R programmers to call *setOLAPRGatewayPort([your port number])*
`LogResultDataSet`	By default OLAPRGateway will not log the data result sets being returned. If you et LogResultDataSet to True the result sets will be logged.

Installing the R package

Windows

From inside R environment.

```
setwd("C:/FolderWherePackageFileIs/");
packages<-dir();
install.packages("olaprgateway_1.0.tar.gz", repos=NULL)
```

Ubuntu and Mac OS X

From a bash shell / terminal window

```
R CMD INSTALL olaprgateway_1.0.tar.gz
```

Using the OLAPRGateway library in R

The OLAPRGateway library contains five methods. setOLAPRGatewayToken is required before subsequent …Frame() request can be made. By default OLAPRGateway library for R will default to use 8888. The R programmer can change the target port by calling setOLAPRGatewayPort(port)

Definitions and examples

setOLAPRGatewayPort(port)

```
setOLAPRGatewayPort(8888)
```

setOLAPRGatewayHost(host)

```
setOLAPRGatewayHost("127.0.0.1")
```

If running OLAPRGateway.exe on the same machine you are doing R programming then set this to 127.0.0.1. If OLAPRGateway.exe is running on a different machine on your network, specify the IP address of that machine here.

setOLAPRGatewayToken(token)

```
setOLAPRGatewayToken("Msuyu2s9")
```

getOLAPFrame(command)

```
result <- getOLAPFrame("SELECT NON EMPTY Hierarchize ({DrilldownLevel ({ [Dim
Property].[Property Map].[All] } , , , INCLUDE_CALC_MEMBERS ) } ) DIMENSION PROPERTIES
PARENT_UNIQUE_NAME , HIERARCHY_UNIQUE_NAME ON COLUMNS , NON EMPTY Hierarchize
({DrilldownLevel ({ [Dim InDma].[In DMA].[All] } , , , INCLUDE_CALC_MEMBERS ) } )
DIMENSION PROPERTIES PARENT_UNIQUE_NAME , HIERARCHY_UNIQUE_NAME ON ROWS FROM [Mobile]
WHERE ( [Measures].[Visits] ) CELL PROPERTIES VALUE")
```

getSQLFrame(command)

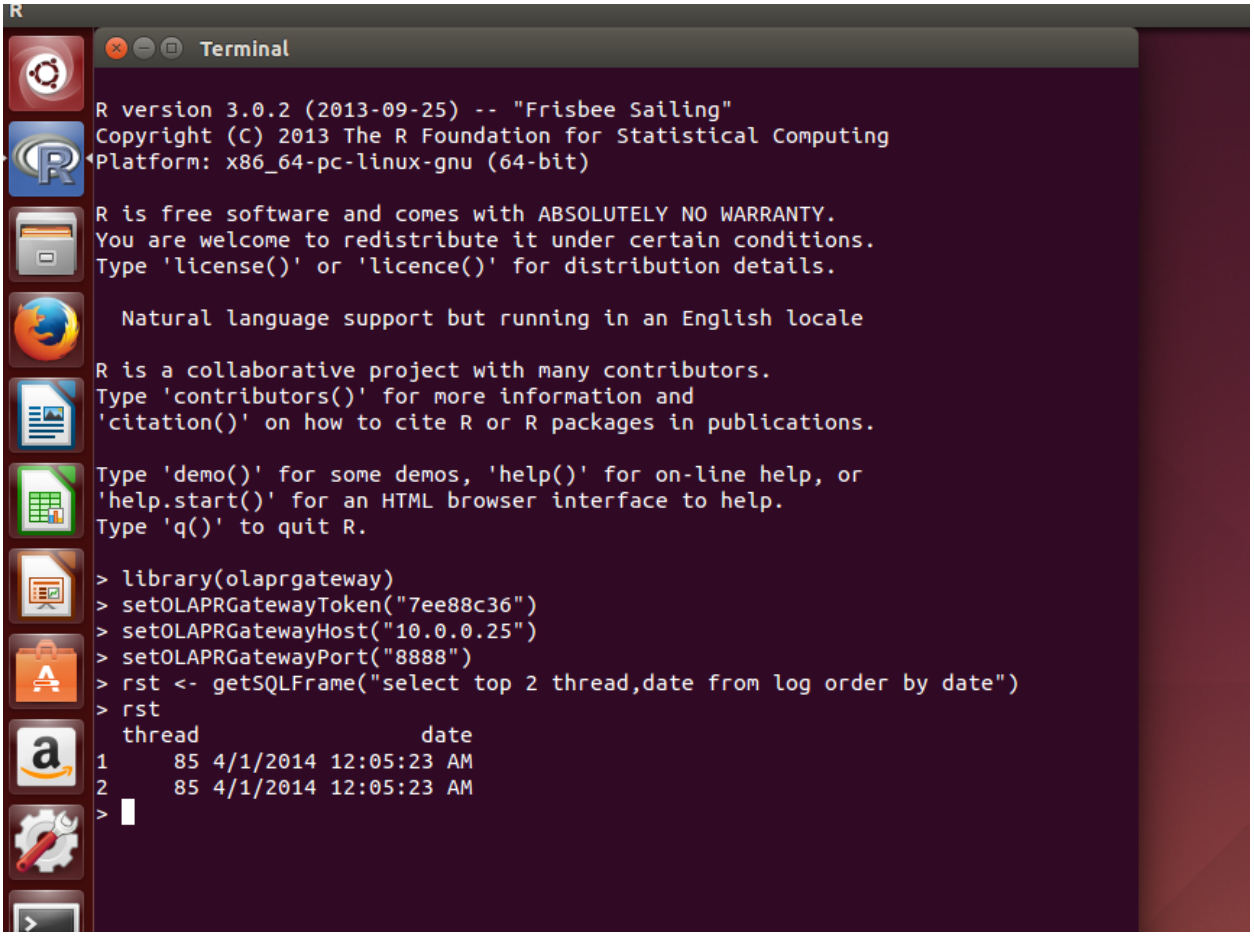
```
result <- getSQLFrame("SELECT AGE,DOB,LNAME,FNAME,ZIPCODE FROM USERS WHERE AGE > 18")
```

getDotFrame(lib,class,command)

```
result < getDotNETFrame("Lib=OLAPRGatewaySamplePlugin.dll;

Class=OLAPRGatewaySamplePlugin.SimplePlugin2;Param=")
```

Example of SQL query performed in R



Developing Plug-ins

In addition to SQL Server and OLAP data sources, OLAPRGateway can be extended to offer custom data providers to R programmers. To develop a custom OLAPRGateway plug-in the .NET developer will need to reference **OLAPRGatewayExt.dll** library and create one or more new classes derived from the **OLAPRGatewayPlugin** abstract base class.

OLAPRGatewayPlugin requires the .NET developer to implement a single method **ReturnDataTable** which must accept a single param string and return an ADO.NET DataTable object. When the method is requested by an R Programmer the OLAPRGateway will transform the DataTable object and return an R complaint data.frame script. The resulting Plugin .dll files should be placed in the **\Lib** subfolder. The **\Lib** subfolder should exist at the same level as OLAPRGateway.exe. When OLAPRGateway.exe is launched it will scan **\Lib** for all available .dll files and attempt to load them into memory. The R developer can then make use of the plug-in though the OLAPRGateway R library by calling getDotNETFrame() like so

Usage

```
result < getDotNETFrame("Lib=OLAPRGatewaySamplePlugin.dll;
Class=OLAPRGatewaySamplePlugin.SimplePlugin2;Param=")
```

Below you'll find example source code for the SimplePlugin2 example. All source required to build this sample is available on GitHub.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using OLAPRGatewayExt;
using System.Data;


namespace OLAPRGatewaySamplePlugin
{
    public class SimplePlugin2 : OLAPRGatewayPlugin
    {
        public override DataTable ReturnDataTable(string param)
        {
            DataTable dt = new DataTable();
            dt.Columns.Add(new DataColumn("ID"));
            dt.Columns.Add(new DataColumn("State"));
            dt.Columns.Add(new DataColumn("Population"));
```

```
            DataRow r = dt.NewRow();
            r[0] = "1";
            r[1] = "NY";
            r[2] = "8002";

            dt.Rows.Add(r);

            DataRow r2 = dt.NewRow();
            r2[0] = "2";
            r2[1] = "CO";
            r2[2] = "6003";

            dt.Rows.Add(r2);

            DataRow r3 = dt.NewRow();
            r3[0] = "3";
            r3[1] = "CA";
            r3[2] = "7200";

            dt.Rows.Add(r3);

            return dt;
        }
    }
}
```

Building OLAPRGateway from the source files

OLAPRGateway, OLAPRGatewayExt and OLAPRGatewaySamplePlugin binaries were all built with Visual Studio 2015 Community Edition. Visual Studio 2015 Community Edition is free to download from Microsoft's website here

https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx

If intending to connect to SSAS you may need additional libraries from Microsoft. If error messages persist please see the following article and reference the section on ADOMD.NET for recommendations on additional Windows libraries that may be required for the build.

https://msdn.microsoft.com/en-us/library/dn141152.aspx#bkmk_ADOMD

Features planned for future releases

- Option to enable concurrent connections "server mode"
- Port OLAPRGateway console project to Windows Service project type
- Provide User Interface to edit configuration values
- New R library methods to set format transformation on date time and Boolean values
- New R library methods for user mapped column headers for getOLAPFrame request

The most current official build and source can always be found here

https://github.com/jdeats76/OLAPRGateway

If you are interested in seeing specific features added that are currently not planned, feel free to submit suggestions to me though GitHub or at Jeremy.deats@gmail.com

Contributing to OLAPRGateway

OLAPRGateway is free to modify, redistribute and “fork” in accordance with the GNU General Public License. If you would like to contribute to this distribution please provide the following.

- Brief summary of your planned contribution
- Your background
- Anticipated timeframe for development
- Contact information

Submit inquires to Jeremy.deats@gmail.com

About the author

Jeremy Deats is Professional Software Developer, Co-Author and Tech Blogger. With focus on Service-Oriented Architecture and best of breed technologies, Jeremy has helped give many companies in the Oil & Gas, Energy and Media sectors a competitive edge through innovative and cost effective solutions.

Jeremy is currently focusing on Mobile Services and Business Intelligence at Hearst Corporation.