TOC

About This Guide

This is an archived copy of the v23 documentation provided for Logi Info v23.3 and its service packs.

Notice: Archived Documentation

This documentation is provided as a courtesy reference for a version of our software that is no longer under active development or support. The information contained herein is offered without warranties of any kind, either expressed or implied, including but not limited to warranties of accuracy, completeness, or fitness for a particular purpose.

While this archived material may assist with understanding historical functionality, please be aware that the software described is no longer maintained at this version level and may contain outdated or inaccurate information. Images may not reflect currently supported modules, support sites, or third party products. This software may not be compatible with current versions of previously compatible third party products.

To access and upgrade to current software solutions and receive ongoing support, please contact our customer support team. They can assist you in migrating to the latest appropriate software version that meets your needs. Our support team is available to help ensure a smooth transition to actively maintained alternatives that provide the functionality and reliability you require.

Datasource Connections

A *datasource connection* is the mechanism used by Logi applications to communicate with datasources through a driver or provider for the purpose of reading and writing data. Selecting and configuring a **Connection** element is the first step in the process of developing most Logi applications; connections are discussed in this topic.

The following topics provide more details about the available datasource connections:

About Datasource Connections

Logi applications connect to a datasource in order to read and write its data. In most cases, this is done using a **Connection** element, which specifies the nature of the communication, including security credentials.

The exceptions are for accessing certain kinds of files, such as XML, CSV, and Excel files, which your Logi application is able to access directly through the web server file system. These sources *do not* use Connection elements.

Connection elements do not work with datasources requiring SSH.

Connection elements are configured in the _Settings definition and are therefore available application-wide. One connection can be used for multiple report definitions, and a Logi application can have multiple connections, allowing data from multiple sources to be included in a single report. There are three types of connection elements:

Vendor-Specific Connections

Vendor-specific connection elements, such as **Connection.MySQL**, are the connections we recommend you use. They provide the best performance and easiest configuration. The available vendor-specific Connection elements include:

- Connection.DB2
- Connection.Google Docs
- Connection.Google Maps
- Connection.MongoDB
- Connection.MySQL
- Connection.OpenEdge
- Connection.Oracle
- Connection.PostgreSQL

- Connection.Redshift
- Connection.Salesforce
- Connection.SQLServer (Microsoft)
- Connection.Sybase
- Connection.Twitter
- Connection.Vertica

Your Logi product, in combination with the .NET framework or Java libraries, contains all of the underlying *drivers* or *providers* necessary to make all of these connections work, except for **Oracle**, **DB2**, and **Sybase.** Due to licensing restrictions, these connections are only usable if you have the client driver software that's distributed with the database server, or that you download. See the *Usage Notes* sections for each of these systems on page two for information about their drivers.

MySQL 5.5 is not supported for use with our *DataLayer.ActiveSQL* technology.

The configuration of **Connection.Google Docs** and **Connection.Google Maps** is discussed in "Google Connections" on page 101.

 Salesforce.com has discontinued support for the TLS 1.0 protocol. If you're using **Connection.Salesforce**, or REST or SOAP API connections to Salesforce, you must be using Logi Info v12.2-SP4 or later, which supports the TLS 1.1 and 1.2 protocols. In addition, Login Java applications must use JDK or OpenJDK 8 or later to use the appropriate protocol version. For more information about which versions of JDK works with Info, see *Java Usage Policy*.

The most recent .NET and Java drivers shipped with Logi Info v12.6 only work with **MongoDB** 2.6 and later. If you're using an older version of MongoDB, such as 2.4, you cannot upgrade your Logi application to Info v12.6.

Generic Connections

Generic connection elements, such as **Connection.OLEDB**, require you to provide, or use a wizard to construct, a "connection string". Connection strings are text strings that are passed in code to an underlying driver or provider in order to initiate the connection. The available generic Connection elements include:

- Connection.ADO
- Connection.JDBC
- Connection.LDAP
- Connection.ODBC
- Connection.OLAP
- Connection.OLEDB
- Connection.SMTP

Connection.OLAP is used to connect to Microsoft Analysis Services OLAP databases and **Connection.SMTP** is used to connect to SMTP email servers.

The other generic elements are for connection to a variety of databases. For example, the **Connection.OLEDB** element allows you to use the Microsoft Jet driver to connect to MS Access database files and can also be configured to connect to Microsoft's SQL Server database server.

 Microsoft elected not to include the MS Jet driver in its 64-bit OS versions, but it is possible to use the "MS Ace" OLEDB driver, distributed with Office 2007, instead.

Special-Purpose Connections

Special-purpose connection elements, such as **Connection.Web Service,** are designed to connect with specific web services or special Logi services. The available special-purpose Connection elements include:

- Connection.DataHub
- Connection.HTTP
- Connection.Leaflet Map
- Connection.Logi Application Service
- Connection.OData
- Connection.REST
- Connection.Scheduler
- Connection.Web Service (SOAP)

Connection.DataHub is used to connect to our Logi DataHub data virtualization product. **Connection.Logi Application Service** is used to connect to Logi Platform Services, installed with Logi Info 12.5+ or the Discovery Module 3.0. **Connection.HTTP** allows the following datalayers to use a connection to an HTTP or HTTPS datasource that requires authentication:

- DataLayer. CSV
- DataLayer.Excel
- DataLayer.Fixed Format File
- DataLayer.GPX File
- DataLayer.JSON
- DataLayer.KML File
- DataLayer.Web Feed
- DataLayer.Web Scraper
- DataLayer.XML

The **Request Header** is a child element of Connection.HTTP and Connection.REST. This element allows custom information, primarily intended for authentication purposes, to be sent through the connection in the HTTP request header.

Connection.Leaflet Map is used to connect to a Leaflet API-compatible map server. See *Leaflet Maps* for more information.

The **Connection.Logi Application Service** element is used to connect to a special service installed with the Discovery Module and is only availablein Logi Info if the add-on moduleis installed. Connection configuration details are available in Develop with the Discovery Module 3.2.

Connection.OData, and its **OData Parameters** child element, allow connection to REST-style APIs.

Connection.REST and **Connection.Web Service** are similar, in that they both connect to web services. However, Connection.REST works with Representational State Transfer (REST) protocols, such as HTTP, and Connection.Web Service works with the SOAP protocol. **Connection.Scheduler** is used with the Logi Scheduler service provided in Logi Info to manage scheduled reporting.

The Studio Connection Element

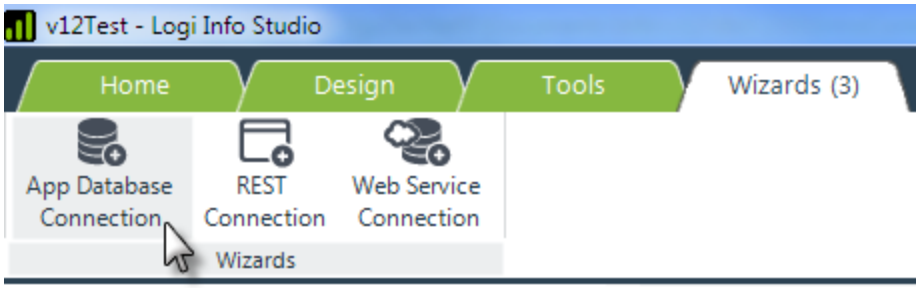
Logi Studio includes tools, like the SQL Query Builder and the Database Browser, that normally use the configured Connection elements to transparently connect to the data during development. However, for a variety of reasons, the Windows development machine where Studio is installed may not be able to use the configured data connections. For instance, the driver may not be installed on, or may be incompatible with, the Windows machine running Studio. In this case, a special **Studio Connection** element can be added to provide a separate connection for Studio's tools. Studio Connection is available as a child of the following connections:

- Connection.DataHub
- Connection.DB2
- Connection.JDBC
- Connection.MySQL
- Connection.Oracle
- Connection.PostgreSQL
- Connection.Redshift
- Connection.SQLServer
- Connection.Sybase
- Connection.Vertica

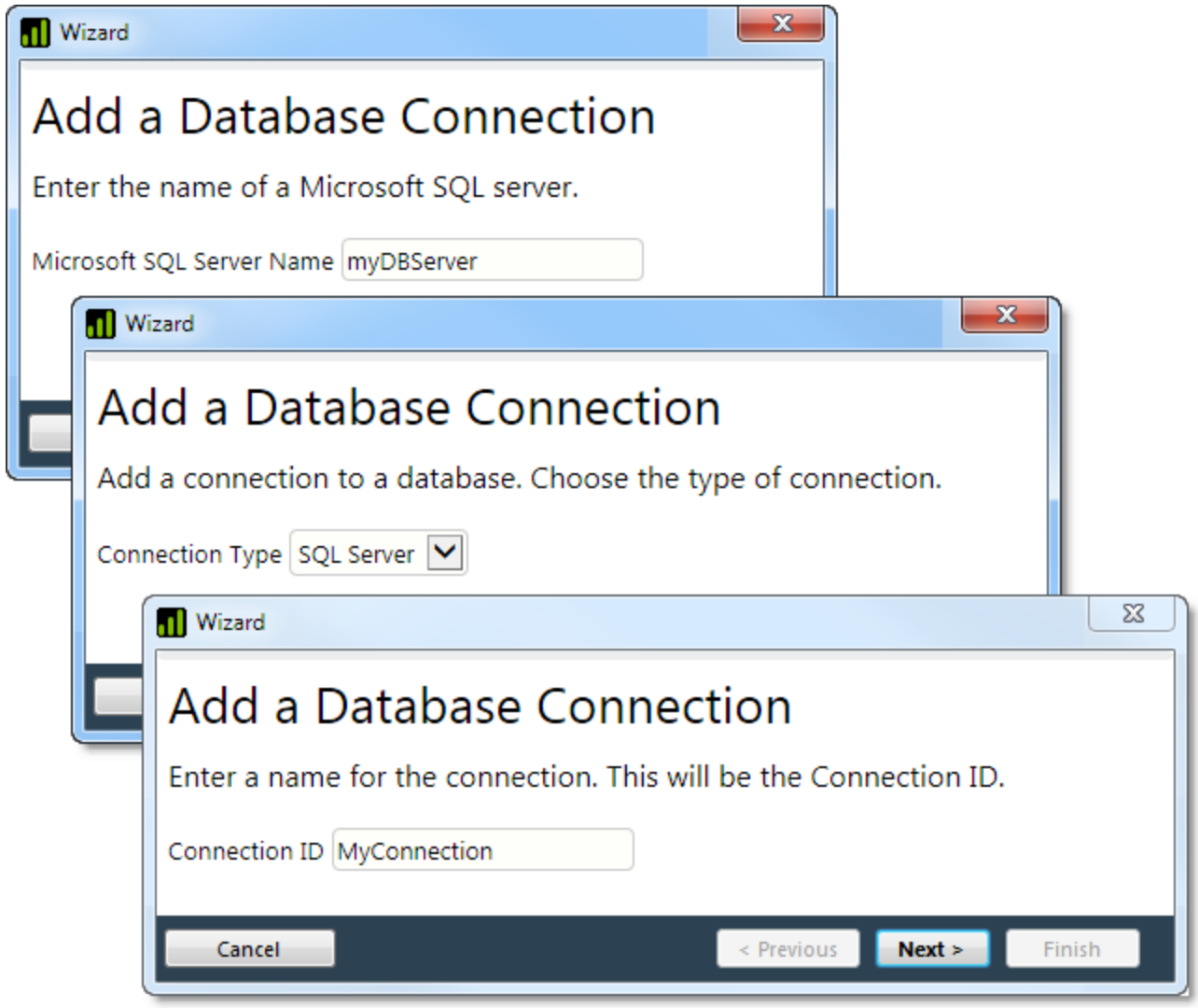
-

Using the Connection Wizard

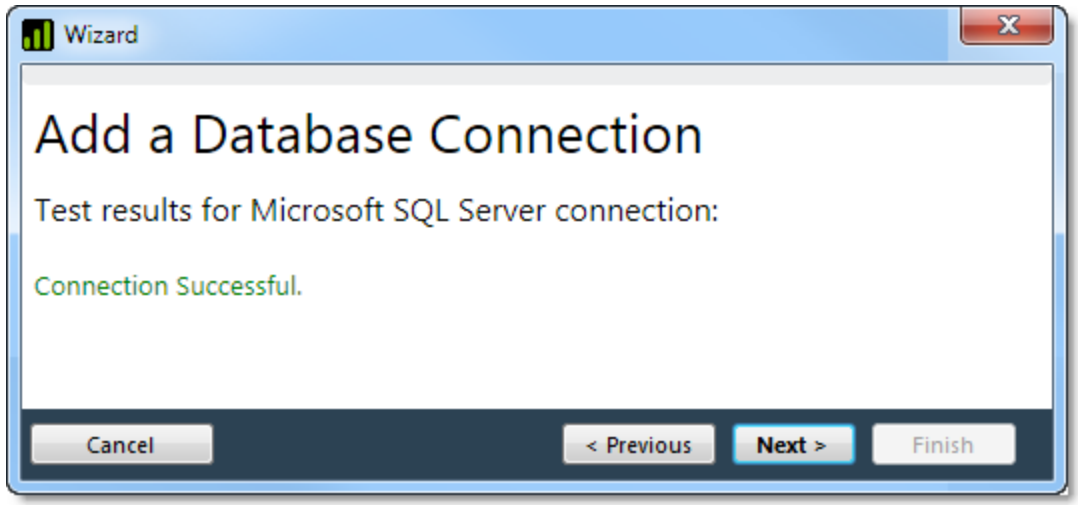
The fastest and easiest way to add and configure a Connection element to your application is to use one of Logi Studio's **Connection Wizards**. Here's how:



1. In the _Settings definition in Logi Studio, select the Connections element and either click a Connection item in the main menu's Wizards tab (shown above) or right-click the element and select *Element Wizards* → *Add a Connection* item from the context menu. For this example, we'll add a Database Connection.



2. A series of dialog boxes, like those shown above, will be displayed. You'll be asked to identify the server, the database, and to provide server login credentials. Provide appropriate information for your datasource and click **Next** to move to the next dialog box.

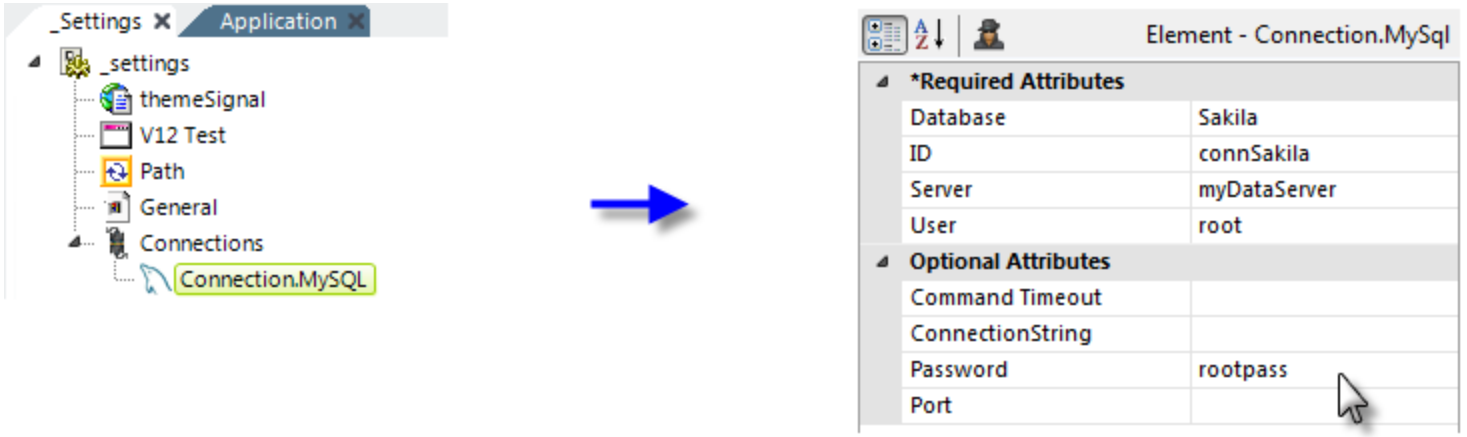


3. Finally, the connection between Studio and the database server will be tested, as shown above. When you click **Next** and **Finish**, an appropriately configured Connection element will be added to the _Settings definition.

That's it - your connection is ready to be used.

Using Vendor-Specific Connections

Vendor-specific Connection elements are designed to make configuring a connection very simple.



As shown in the example above, a **Connection.MySQL** element has been added to the _Settings definition in an application. Its attributes allow the developer to specify the server name, database name, and access credentials. At runtime, the required connection string will be created behind-the-scenes using this information.

Optional parameter elements, as shown above, can be added beneath the connection element. These provide the flexibility of including additional connection string parameters, if necessary. For example, the optional parameters for the MySQL connection include Compress, Connection Lifetime, Direct, Embedded, Port, Protocol, and Unicode. You can also enable special MySQL connection behavior, like use of User Defined Variables, by creating the parameter Allow_User_Variables and setting it to *True*.

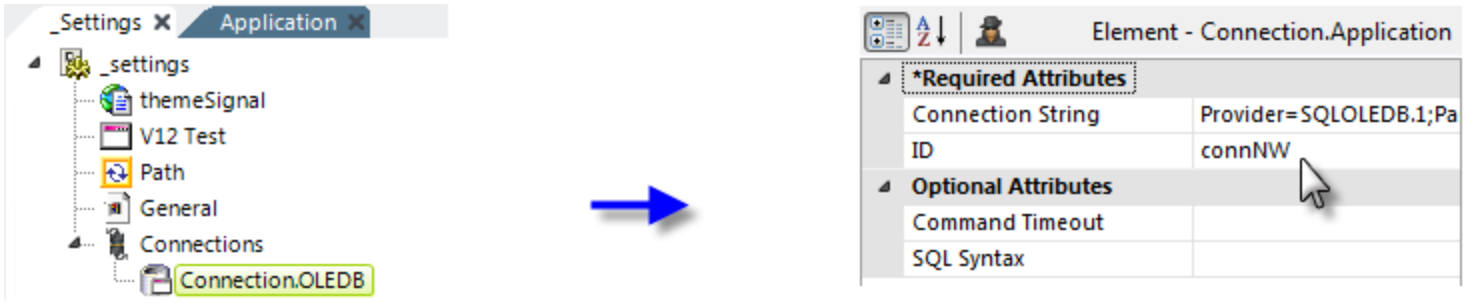
The other attributes for Connection.MySQL and similar connection elements, are:

Attribute	Description
ID	(Required) Specifies a unique element ID; referred to by otherelements that use this connection.
Command Timeout	Specifies the amount of time, in seconds, before the request to connect to the datasource is presumed to have failed. For most datasources, the default value is *60* seconds. For MySQL, the default is *30* seconds.
Connection String	A fully-formed connection string for the database. If a value is defined here, all other attributes will be ignored and this string will be used to connect to the database. Any child parameter elements will be added to the connection string.
Port	The port number for the database, if the default number is not being used.

In general, vendor-specific connections are much easier to configure than generic connection and developers should look to them first when creating applications.

Using Generic Connections and Connection Strings

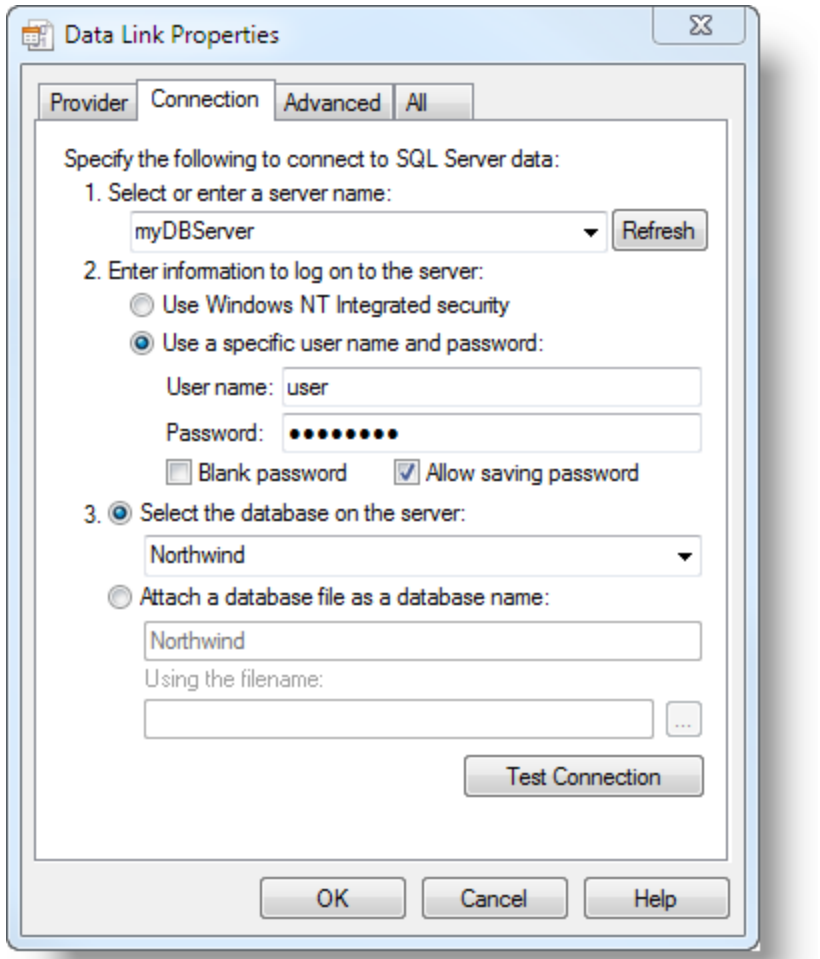
Another method of creating a connection is to use one of the *generic* Connection elements, which requires that you provide a connection string.



In the example shown above, a **Connection.OLEDB** element has been added to the _Settings definition. Its **Connection String** attribute has been configured with a connection string, which looks like this in its entirety:

```
Provider=SQLOLEDB.1;Password=password;Persist Security Info=True;User ID=user;Initial Catalog=Northwind;Data Source=myDBServer
```

You can see that the connection string has a specific format, which is unique to the connection type. The Connection.OLEDB element can invoke a special wizard to build its connection string; for all other connections, developers will need to provide an appropriate string.



The **Data Link Properties** wizard, shown above, assists in the creation of connection strings for the Connection.OLEDB element. In Studio, click the little browse button at the end of the element's Connection String attribute value to invoke the wizard. The **Test Connection** button only ensures network connectivity to the database server - it *does not* guarantee that the User

Name and Password entered are valid! The test can be successful and yet you may still not be able to access the data if the credentials are wrong. If providing specific credentials, be sure that you enter them carefully and check *Allow saving password*.

This very useful web site: http://www.ConnectionStrings.com provides example connection strings for a wide variety of datasources and can assist in providing the proper format for strings for other connection types.

With ActiveSQL

If **Connection.JDBC** or **Connection.ODBC** is used with DataLayer.ActiveSQL, you must configure the Connection element's optional **SQL Syntax** attribute to identify the database's SQL syntax version.

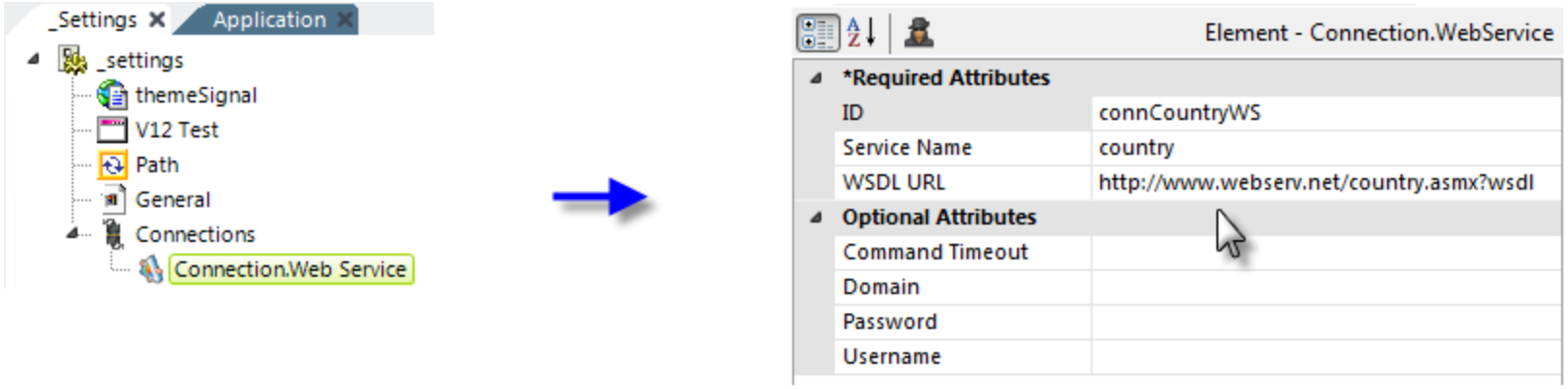
Notes For Java Developers

When using a **Connection.JDBC** element, specifying an IP address instead of a host name in the connection string may significantly *increase* the connection opening time, due to the underlying Java implementation. Therefore, it is recommended that developers use host names, *not* IP addresses, in connection strings to minimize the time it takes to open a connection.

The **Connection.SMTP** element supports TLS/SSL authentication in Java applications only. This allows connection to Gmail and other similar services that will programmatically send email on your behalf. Set the element's **SMTP Authentication Mode** attribute to *3* to select TLS/SSL. In v12.2-SP4+, the TLS 1.1 and 1.2 protocols are supported.

Using Special-Purpose Connections

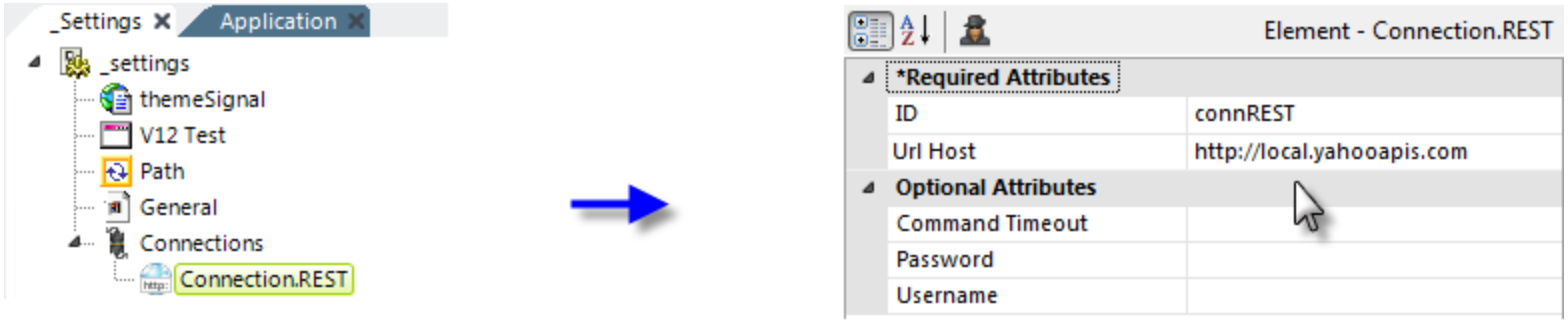
Special-purpose connections are used to connect to specific web services or to special OS services for the Logi Scheduler and other facilities. All require credentials to authenticate the connection.



In the example shown above, a **Connection.Web Service** element has been added to the _Settings definition. The attributes identify specific aspects of the web service. As with all connections, a unique ID is required so that it can be referenced later by datalayer elements.

For developers connecting to their *own* web service, the WSDL document specified in this connection element's required WSDL URL attribute must be "valid", meaning that it can contain no errors that would cause it to fail to compile. A document that's readable but that will not compile is invalid and will cause the connection to fail.

Connection.Web Service has an optional child element, **Connector Property Parameters**, which makes it possible to connect to a web service which requires special authentication and proxy settings.



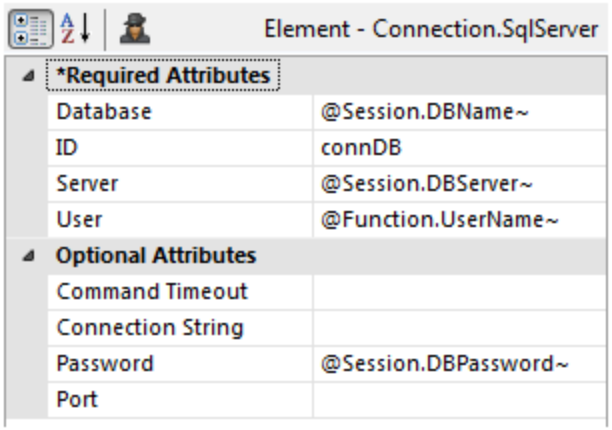
In the next example, shown above, a **Connection.REST** element has been added to the _Settings definition. Its **Url Host** attribute identifies the web service host address, and other attributes are used to provide user credentials, if required. As with all connections, a unique ID is required so that it can be referenced later by datalayer elements.

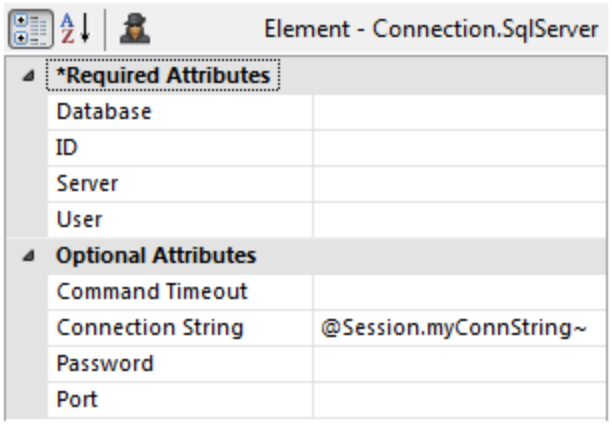
Connection.REST with Windows Domain Authentication

The Connection.REST element includes an attribute that lets you use Windows domain authentication, also known as "integrated security", with your connection. To enable this, set the connection element's **IntegratedSecurity** attribute to *True* and leave the **Username** and **Password** attribute values blank. The integrated security credentials that the application is running under will be used for the connection.

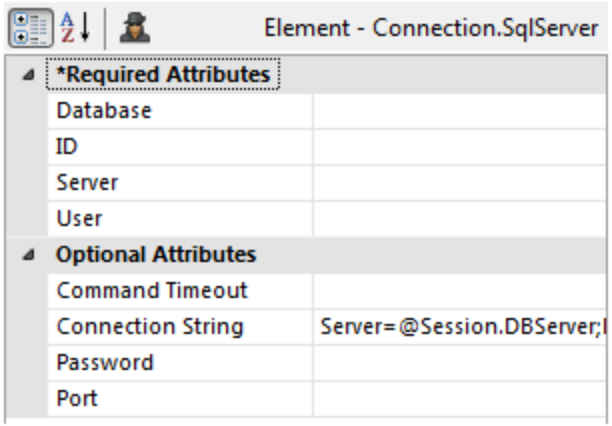
Creating Dynamic Connections

You may want to create *dynamic* connections, which connect to a datasource based on external criteria. Connection elements make this possible by accepting tokens in their attributes.





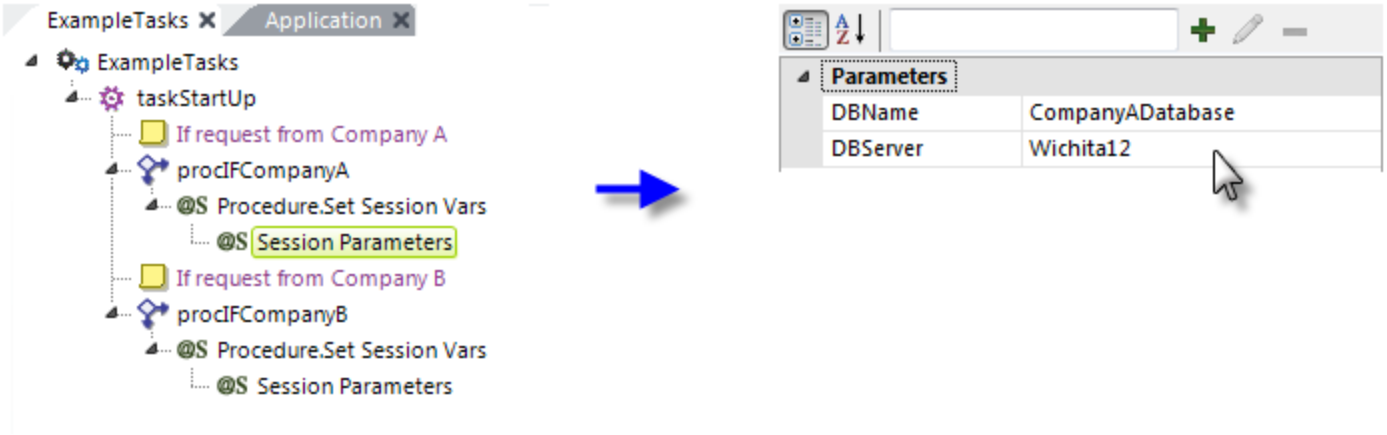
In the example above, left, tokens are used for the required attributes of the **Connection.SQL Server** element. Another approach is to assign a complete connection string to a Session variable and use its token in the **Connection String** attribute, as shown above right. A value in that attribute will cause the "required" attributes to be ignored.



Or you can enter the literal connection string text into that attribute, as shown above, and tokenize parts of it, like this:

```
Server=@Session.DBServer~;Database=@Session.DBName~;User Id=@Function.UserID~;Password=@Session.DBPassword~;
```

There are a variety of techniques available for setting the token values prior to the processing of the _Settings definition. For example, the Startup Process element, when included in _Settings, can be used to call a Process task that assigns the Session variables needed. This task will run *before* the Connection elements are processed. For more information, see *The _Settings Definition*.



The example shown above is simplistic but shows you how to use a Process task to set Session variables, which can be used later as tokens in the Connection element's attributes. If you're unable to use the Startup Process element for some reason, you can still make the Process task run by calling it directly, rather than calling the default report definition:

```
http://yourServer/yourLogiApp/rdProcess.aspx?rdProcess=yourProcessDefinition&rdTaskID=taskStartUp
```

Connecting with Metadata

Metadata files can be used with the Analysis Grid and Active Query Builder elements to provide a dynamic data environment for end users. You create Metadata files are using a wizard in Logi Studio during application development and they can be highly tailored to present desirable data objects and relationships. Numerous data qualities, including formatting, security restrictions, and joins, can be specified.

The **Metadata** element is used to identify the metadata files to be included with a data connection.

The following Connection elements can use the Metadata element:

- Connection.ADO
- Connection.DataHub
- Connection.DB2
- Connection.JDBC
- Connection.MySQL
- Connection.ODBC v23.1
- Connection.OpenEdge
- Connection.Oracle
- Connection.PostgreSQL
- Connection.RedShift
- Connection.SQLServer
- Connection.Vertica

The original wizard *has been deprecated*. In its place, the "Web Metadata Builder" on page 136 appears in a window within Studio.

1010data

1010data provides a cloud-based software platform and associated services for business analytics and database publishing of large data sets. Here are the installation and configuration instructions for using 1010data with your Logi applications:

Required Software

The following components are required for use of CData's ADO.Net driver for 1010data:

- **Microsoft Visual C++ Redistributable** - This is a CData driver dependency. Available for download at https://www.microsoft.com/en-gb/download/details.aspx?id=13523.
- **1010data .NET SDK** - This is the SDK the CData driver relies upon and is available for download directly from 1010data at https://www.1010data.com/technology/technical-interfaces/software-development-kits/

  On the web page, select the .NET SDK -> 64/32-bit Windows (ZIP) file download; when you expand the .zip file, you should find the "dotNET_2016022" version folder. This is the version *required* for use with the CData driver.
- **Logi Info 64-bit v12.1 SP4+** - Use the latest Logi Info release that supports functionality for 1010data-specific functions. Not compatible with 32-bit versions.
- **CData Driver** - This is an installer file for the CData driver, which is provided separately by Logi Analytics.

Installation for v12.2+

Install the required software in the following order:

1. Install the Visual C++ Redistributable.
2. Install the CData Driver.
3. Install, or upgrade to, Logi Info 64-bit 12.1.SP4+

4. Create a new Logi Info application, or upgrade an existing Info application, to the newly installed version of Logi Info.
5. Close Logi Studio, if it's currently running.
6. Locate and copy these files (assumes a default Logi Info installation location):

   `C:\Program Files\LogiXML IES Dev\Engine64\12.2.116\CData.C1010.System.dll`

   `C:\Program Files\LogiXML IES Dev\Engine64\12.2.116\System.Data.CData.C1010.dll`

   to this folder:

   `C:\Program Files\LogiXML IES Dev\LogiStudio\bin`
7. Expand the 1010data .NET SDK .zip file and locate and copy this file:

   `dotNET_2016022\win64\CppCLISDK1010.dll`

   to this folder:

   `C:\Program Files\LogiXML IES Dev\LogiStudio\bin` (assumes a default Logi Info installation location)
8. Stop the IIS web server.
9. Copy the `dotNET_2016022\win64\CppCLISDK1010.dll` file, from Step 7, to the folder *<your Logi Application folder>*`/bin`.
10. Restart the IIS web server.

Initial installation and configuration is complete. When creating additional applications, the steps for copying the two .dll files will have to be completed for each new application.

Installation for versions prior to v12.2

Install the required software in the following order:

1. Install the Visual C++ Redistributable.
2. Install the CData Driver.
3. Install, or upgrade to, Logi Info 64-bit 12.1.SP4+
4. Create a new Logi Info application, or upgrade an existing Info application, to the newly installed version of Logi Info.
5. Close Logi Studio, if it's currently running.

6. Locate and copy this file:
   `C:\Program Files\CData\CData ADO.NET Provider for 1010Data 2016\lib\4.0\x64\System.Data.CData.C1010.dll`
   to this folder:
   `C:\Program Files\LogiXML IES Dev\LogiStudio\bin` (assumes a default Logi Info installation location)
7. Expand the 1010data .NET SDK .zip file and locate and copy this file:
   `dotNET_2016022\win64\CppCLISDK1010.dll`
   to this folder:
   `C:\Program Files\LogiXML IES Dev\LogiStudio\bin` (assumes a default Logi Info installation location)
8. Stop the IIS web server.
9. Copy the `System.Data.CData.C1010.dll` file, from Step 6 to the folder *<your Logi Application folder>*`/bin`.
10. Copy the `dotNET_2016022\win64\CppCLISDK1010.dll` file, from Step 7, to the folder *<your Logi Application folder>*`/bin`.
11. Restart the IIS web server.

Initial installation and configuration is complete. When creating additional applications, the steps for copying the two .dll files will have to be completed for each new application.

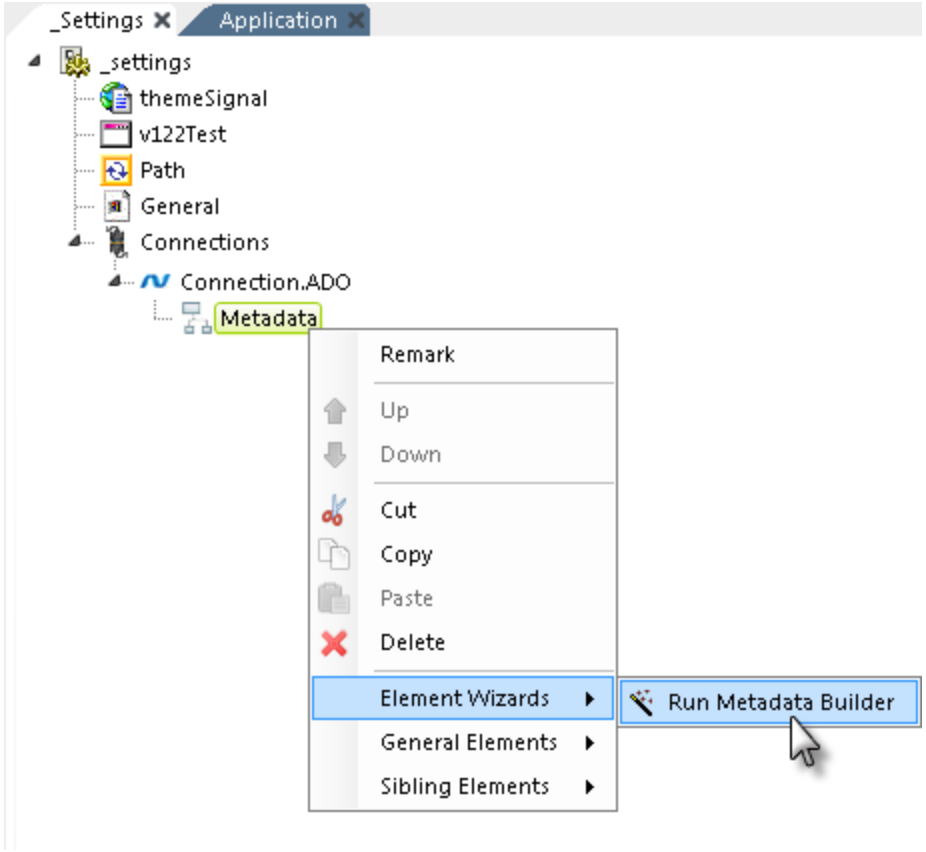
Using the Connection.ADO Element

The Connection.ADO element is used, in the _Settings definition, to connect to 1010data. Its attributes are:

Attribute	Description
ADO Connection String	(Required) The a connection string valid for the ADO.NET data provider. Valid parameters include: **User** - Specifies the username to be used for login. **Password** - Specifies password associated with the username.

Attribute	Description
	LoginURL - Specifies the URL (or gateway) to the server. **Login Type** - Specifies the type of login to use. Options include *Possess*, *Kill*, or *NoKill*. Defaults to *Possess*, which will not interrupt sessions if you are also logged in via the browser. **Group** - Specifies the group to log in as (used for pooled connections). You must set the User to the name of a group admin. **QueryPassthrough** - Specifies a boolean which indicates if the query will contain 1010data XML to be executed. Default value: *false* **UseSimpleNames** - Specifies a boolean that determines whether or not names will be simplified to alpha-numeric with underscores. Specifying *false*, for example, will make Data Tables display as they appear in 1010data. Default value: *true*Example connection string: `user=someUserID;password=myPass;LoginURL=https://www2.1010data.com/cgi-bin/x.xx/ab.c;`
Assembly Name	(Required) Specifies the provider assembly name. For Logi apps, specify this value: *System.Data.CData.C1010.dll*
ID	(Required) Specifies a unique, arbitrary connection element ID.
Command Timeout	Specifies the amount of time, in seconds, before the request to connect to the data source is presumed to have failed. Default value: *60* seconds.
SQL Syntax	Specifies the type of database server being queried so that the proper syntax is used. For Logi apps, specify this value: *1010data*

Once the connection is configured, we recommend that you create metadata by adding a child **Metadata** element beneath the connection element.



Then, as shown above, you can configure the query and metadata using the Metadata Builder wizard.

Amazon Redshift

Amazon Redshift is a fast, fully-managed, petabyte-scale data warehouse service in "the cloud". It's optimized for datasets ranging from a few hundred gigabytes to a petabyte, or more. Before using the service, you must sign-up and be licensed by Amazon. For more information, see the Amazon Redshift website.

The **Connection.Redshift** element allows you to connect your Logi application to the Amazon service. This is an ODBC connection and requires an ODBC connection string that includes the credentials assigned to you by Amazon.

For a .NET application, you'll need to install the Amazon Redshift ODBC Driver (32- or 64-bit, matching your Logi Info installation), and configure it with appropriate settings and a connection string with account credentials. This process is described in detail in this Amazon Web Services document.

For earlier versions of Logi Info .NET and Info Java applications, you must use the PostgreSQL 8.x JDBC and ODBC drivers to ensure compatibility. The PostrgreSQL 9.x JDBC and ODBC drivers might not work properly with all applications.

Though you can have success using standard DataLayer.SQL, for best performance with very large datasets, we recommend use of *DataLayer.ActiveSQL*. Amazon Redshift SQL is based on PostgreSQL 8.0.2. syntax but has a number of very important differences that you must be aware of as you design and develop your data warehouse applications. Please refer to the Amazon SQL Reference for more information.

Because of the nature of the Amazon Redshift service, Studio's wizards, Query Builder, Database Browser, and the "available columns" list in the Attributes panel, may not work, or work completely,in all cases.

Connection.ADO

Connection.ADO enables data connections using ADO drivers, such as HPCC from CData. These connections are normally used with DataLayer.SQL elements. See the Using the Connection.ADO Element section in the "1010data" on page 31 topic for a detailed description of its attributes. The following Connection.ADO example works with HPCC clusters:

<Connection Type="ADO"
AdoAssembly="System.Data.CData.HPCC.dll" ID="connHPCC" AdoCon-
nectionString="URL=http://192.168.nnn.nnn:8510;Offline=false;Cluster=*yourCluster*;Version=1" CommandTimeOut="30" />
And here's an example of using it to connect to Salesforce :

<Connection Type="ADO"
AdoAssembly="System.Data.CData.Salesforce.dll" ID="connSalesforce"
AdoConnectionString="user="*yourEmailAddress*;password=*yourSFPassword*;security token=*yourSFSecurityToken*;"/>

ADO connections work with Logi .NET applications only; they are not usable in Java applications. This connection element also allows you to use ADO-style drivers that you may develop yourself, however, Logi does not guarantee your driver will work.

Connection.JDBC and SQL Server 2008

Developers creating Logi applications for Java, and using Connection.JDBC to connect to Microsoft SQL Server 2008, may encounter the following errors: `There was a problem running a DataLayer. The error was: Could not open the supplied connection to the data. The server version is not supported. The target server must be SQL Server 2000 or later.` This occurs when a newer JDBC driver is required for SQL Server 2008. The solution is:

1. Download a new driver from Microsoft and uncompress the download.
2. From the download, copy the file `sqljdbc4.jar` to your Logi application's `WEB-INF/lib` folder.
3. Delete the old driver, `sqljdbc.jar`, from the same folder.
4. Restart the web server.
5. Run the Logi application.

If the following error message is received: `Java Runtime Environment (JRE) version 1.6 is not supported by this driver. Use the sqljdbc4.jar class library, which provides support for JDBC 4.0.` it means you either did not remove the original `sqljdbc.jar`, replaced it with the `sqljdbc.jar` file that came with the download instead of adding `sqljdbc4.jar` to the folder, or did not restart the web server.

Configuring Connection.LDAP

The Connection.LDAP element has the following attributes:

Attribute	Description
Base DN	(Required for Java, ignored for .NET) Specifies the top level of the LDAP structure, its "base Distinguished Name". The attribute value is derived from the company DNS domain name components. For example, to connect to an LDAP server within the domain www.MassiveDynamic.com, the attribute value would be this string: `dc=MassiveDynamic,dc=com`
ID	(Required) The unique element ID.
Server	(Required) Specifies the LDAP server host machine name.
Command Timeout	Specifies the amount of time, in seconds, before the request to connect to the data source is presumed to have failed. Default value: *60*
Connection String	Specifies a complete connection string to the LDAP server. If a value is provided, it will be used instead of the connection string generated by the element and all other attributes will be overridden.
Password	Specifies the password for the LDAP user Distinguished Name (DN).
Port	Specifies the port address of the LDAP server. Default value: *389*.
Server Type	Specifies an LDAP server type. Specify *Standard LDAP* for open system servers, such as OpenLDAP, Apache Directory, OpenDS, etc. Specify *MS Active Directory* for Windows Active Directory. Default value: *Standard LDAP*

Attribute	Description
User DN	Specifies an LDAP user "Distinguished Name" and must represent a user than can query the LDAP server. For example, the attribute value would be this string: `uid=john.smith,ou=Users,dc=MassiveDynamic,dc=com`

Database Browser and Query Builder Connections

The Database Browser and Query Builder tools *will* work with the databases for which we provide "native" Connection elements, such as Oracle*, MS SQL Server, and MySQL. They *may* work with other databases that use a connection that mimics that of a database for which we provide a native connection, such as PostgreSQL. They are not guaranteed to work with every possible database and, in that case, we recommend that you use tools provided with the database to examine and manipulate data, and create and test queries, outside of Studio.

* Due to the way in which Oracle configures its software, the Database Browser and Query Builder will *not* connect, using our native Connection.Oracle element, to Oracle databases when Logi Studio is run on a 64-bit platform. In this situation, developers can choose to do without these tools or to use Connection.ODBC to connect.

HP Vertica

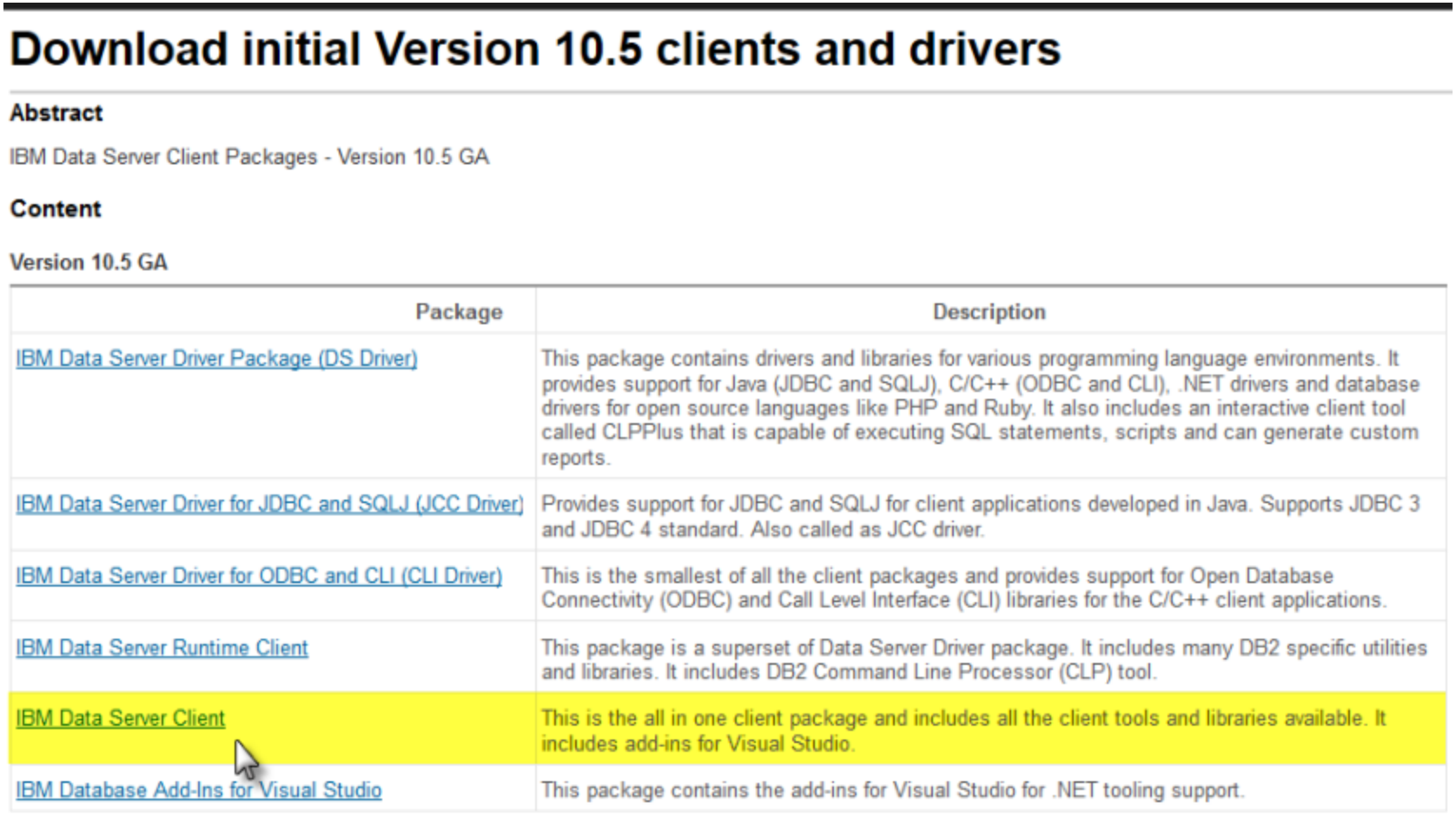
The column-oriented Vertica RDBMS is designed to manage large, fast-growing volumes of data and provide very fast query performance when used for data warehouses and other query-intensive applications. It claims to drastically improve query performance over traditional relational database systems, provide high-availability, and petabyte scalability on commodity enterprise servers. You can download a free, feature-limited copy for local use, or subscribe to their hosted service. For more information, see the Vertica website.

The **Connection.Vertica** element allows you to connect your Logi applications to the database. Special drivers must be downloaded and installed. For more information, see "Working with HP Vertica" on page 185.

Though you can have success using standard DataLayer.SQL, for best performance with very large datasets, we recommend use of *DataLayer.ActiveSQL*. Vertica SQL is based on standard SQL syntax but has a number of very important differences that you must be aware of as you design and develop your data warehouse applications. Please refer to the Vertica SQL Reference for more information

IBM DB2 Universal

As mentioned earlier, the data provider for DB2 is *not* distributed with Logi Info. However, you can download the provider from the IBM software site for use with your Logi application. You'll need to have an IBM ID to download files from the site which, as an IBM customer, you may already have.



Download initial Version 10.5 clients and drivers

Abstract

IBM Data Server Client Packages - Version 10.5 GA

Content

Version 10.5 GA

Package	Description
IBM Data Server Driver Package (DS Driver)	This package contains drivers and libraries for various programming language environments. It provides support for Java (JDBC and SQLJ), C/C++ (ODBC and CLI), .NET drivers and database drivers for open source languages like PHP and Ruby. It also includes an interactive client tool called CLPPlus that is capable of executing SQL statements, scripts and can generate custom reports.
IBM Data Server Driver for JDBC and SQLJ (JCC Driver)	Provides support for JDBC and SQLJ for client applications developed in Java. Supports JDBC 3 and JDBC 4 standard. Also called as JCC driver.
IBM Data Server Driver for ODBC and CLI (CLI Driver)	This is the smallest of all the client packages and provides support for Open Database Connectivity (ODBC) and Call Level Interface (CLI) libraries for the C/C++ client applications.
IBM Data Server Runtime Client	This package is a superset of Data Server Driver package. It includes many DB2 specific utilities and libraries. It includes DB2 Command Line Processor (CLP) tool.
IBM Data Server Client	This is the all in one client package and includes all the client tools and libraries available. It includes add-ins for Visual Studio.
IBM Database Add-Ins for Visual Studio	This package contains the add-ins for Visual Studio for .NET tooling support.

The download page is shown above (the version shown may be different from your version). Download the **IBM Data Server Client** and install it on your web server. This will install some files into the `Windows\assembly` folder and you're all set for Logi .NET applications. For Logi Java applications, once the client package is installed, you'll need to copy the file `db2jcc.jar` from the installation folder to your Logi application's `WEB-INF/lib` folder.

IBM DB2 on iSeries

In order to use a .NET Logi application with DB2 on the IBM AS/400 iSeries platform, you'll need to install some additional software and make a special Connection.DB2 element configuration, as follows:

1. Logi Info uses the IBM Toolkit JT AS400 driver for Java. The file `jt400.jar` has to be downloaded from http://-sourceforge.net/projects/jt400 and placed in the Logi application's `WEB-INF/lib` folder.

2. IBM iSeries Client Access must also be installed and can be downloaded from http://www-03.ib-m.com/systems/power/software/i/access/windows_sp.html.

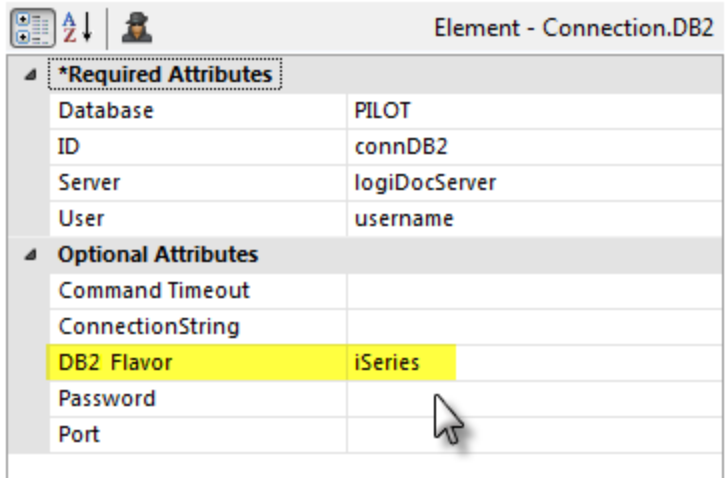
IBM i Access for Windows

(5770-XE1) Release level	Latest Service Pack PTF number	Server maintenance	Date PTF was available	Installed file date	Target for next Service Pack
7.1	SI66062	No co-req PTFs	November 10, 2017	November 18, 2017	June 2018

The current release at the time of this writing is shown above.

3. After installation, the file `C:\Program Files (x86)\IBM\Client Access\IBM.Data.DB2.ISeries.dll` should be copied to the Logi applications `\bin` folder.

4. In your Logi application's _Settings definition, you must configure the Connection.DB2 element to work with the iSeries:



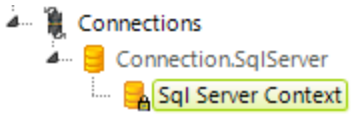
This is done, as shown above, by setting the **DB2 Flavor** attribute to *iSeries*.

Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. It's available in a variety of editions and in local, remote, and cloud-based configurations. It supports a wide variety of transaction processing, business intelligence, and analytics applications in corporate IT environments and is one of the market-leading database technologies.

Context Switching

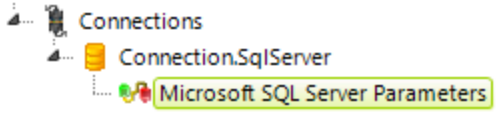
Microsoft SQL Server supports "context switching" - changing of the identity against which permissions to execute statements or perform activities are checked. This is analogous to using the "EXECUTE AS <*UserName*>" command in SQL.



To switch contexts in a Logi application, use a child **SQL Server Context** element beneath the Connection.SQL Server element, as shown above, and set its **Username** attribute value to a valid SQL Server user name, or to any Logi token containing one. The context is re-evaluated with every data request, gets set just before the SQL query is run, and gets reset right after it runs, providing a way to dynamically set the context.

Connection Parameters

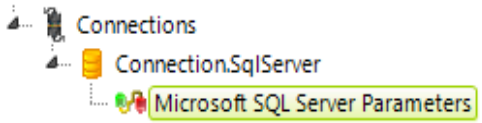
The **Microsoft SQL Server Parameters** element allows you to supply additional parameters when connecting to a Microsoft SQL Server database.



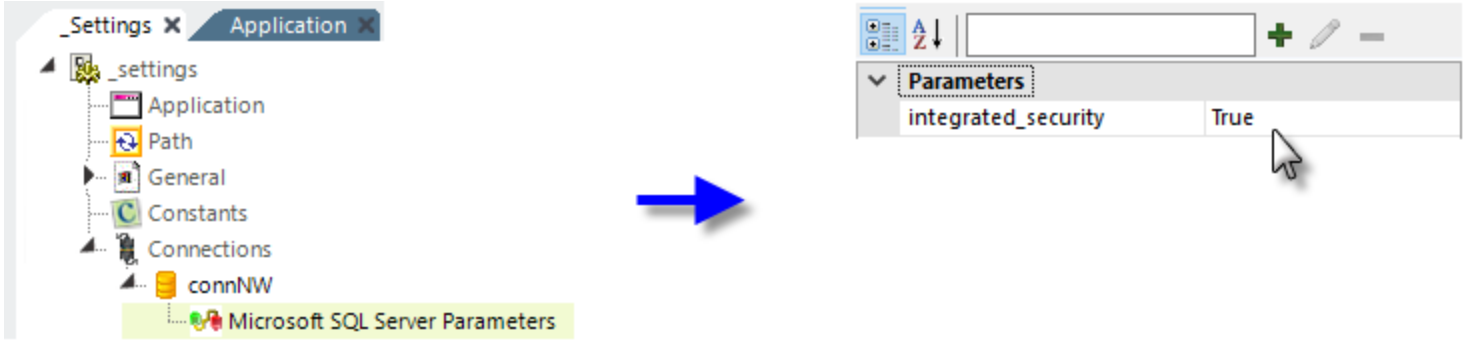
Examples of additional connection parameters include *Async*, *Encrypt*, *Integrated Security*, *Persist Security Info*, *Replication*, and *TrustServerCertificate*.

SQL Server Connection using Windows Authentication

If you're working within a Windows Domain, you may wish to use the domain login credentials as SQL Server login credentials. This may be a convenience just for developers or may be the method you want to employ for all application users. Here are the steps to achieve this:



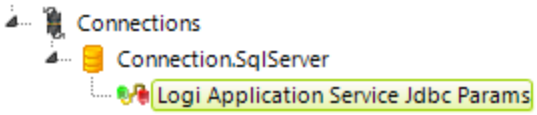
1. Add a **Connection.SqlServer** element, withchild **Microsoft SQL Server Parameters** element, as shown above. Leave the Connection element's **User** and **Password** attributes blank.



2. Configure the Microsoft SQL Server Parameters element to have one parameter, as shown above. *Use the exact spelling and case shown*.

Logi Application Services Parameters

The **Logi Application Service Jdbc Params** element allows you to supply additional parameters when connecting to the Logi Application Service.



Logi Info generates special JDBC connections when the Logi Application Service has been installed. Use this to element to supply additional parameters for connecting to it. An example of a JDBC parameter is *integratedSecurity*.

MongoDB

MongoDB is a cross-platform, document-oriented database system, classified as a "NoSQL" database. Rather than using a traditional table-based, relational DB structure, it uses a collection of JSON-like documents with dynamic schemas. You can download a free, feature-limited copy for local use, or subscribe to their hosted service. For more information, see the MongoDB website.

The **Connection.MongoDB** element allows you to connect your Logi application to the MongoDB server or service. Several special Mongo datalayer elements allow you to retrieve data. For more information, see "Working with MongoDB" on page 193.

The most recent .NET and Java drivers shipped with Logi Info v12.6 only work with MongoDB 2.6 and later. If you're using an older version of MongoDB, such as 2.4, you cannot upgrade your Logi application to Info v12.6.

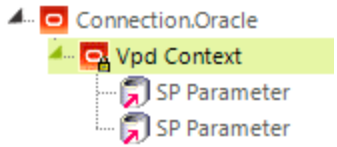
Because of the nature of the MongoDB service, Studio's wizards, Query Builder, Database Browser, and the "available columns" list in the Attributes panel, may not work, or work completely, in all cases.

Oracle

As mentioned earlier, the data provider for Oracle is *not* distributed with Logi Info. Use this link to download Oracle Instant Client drivers (navigate to the appropriate version). Logi Info and SSRM now support Oracle 19c databases.

Virtual Private Database Support

Oracle's Virtual Private Database (VPD) feature enables users to create security policies to control database access at the row and column level, and Logi Info supports this with a special element:



The **Vpd Context** element, highlighted above, is a child of the Connection.Oracle element. The Vpd Context element has a single Command attribute, which is typically set to the name of a Stored Procedure (SP). This SP is set up by an Oracle DBA to create security policies that control database access. Parameters are supplied to the SP by using one or more child SP Parameter elements, as shown above.

"Oracle x.x.x Client Required" Error

Oracle connections may fail, displaying the Inner Error message "System.Data.OracleClient requires Oracle client software version 8.1.7 or greater." in the Debugging Trace Page. This may occur even though the SQL Query Builder is able to connect to the database, and even though a later version of the Oracle Client has been installed.

A likely cause of this problem is that the account that the web server uses to run your Logi application does not have appropriate **fileaccess permissions** to the folder that contains the Oracle Client software. That account may be ASPNET, Network Service, an Application Pool account, or another account, depending on your web server configuration. Giving that account *Read* and *Execute* permissions for the folder that contains the Oracle client will usually resolve this problem.

The web server must be rebooted after the Oracle client software has been installed.

Oracle 32-bit Software on Windows 64-bit OS

If you're attempting to connect to an Oracle database from a Windows platform using one of the following programmatic interfaces: ODBC, OLEDB, OO4O, or ODP.NET, after installing 32-bit Oracle client software on a 64-bit Windows operating system (OS) you may receive one of the following errors: `ORA-12154: TNS:could not resolve the connect identifier specified` `ORA-6413: Connection not open.` This is because the 64-bit Windows OS installs 32-bit software by default into a folder beneath `C:\Program Files (x86)` and the presence of parenthesis in the file path is unacceptable to the Oracle software. This is a known bug that Oracle may fix in the future. Until then, you must install the Oracle 32-bit client software somewhere other than the default location.

SAP Sybase

As mentioned earlier, the data provider for Sybase is *not* distributed with Logi Info. You must have installed the ADO.NET driver from Sybase, which is typically included with your Sybase installation software, or can be downloaded from the Sybase web site.

Connect to MS SQL Server Analysis Services

Logi Info includes dedicated OLAP-related elements that allow users to retrieve and work with data from Microsoft SQL Server Analysis Services (SSAS), also known as Microsoft Analysis Services. The first step in using Logi Info with OLAP data is to *connect* to SSAS; this document discusses several methods of making that connection.

The following topics discuss connecting to SSAS:

- Configuring IIS Authentication
- Configuring the Connection Element

Logi OLAP has been deprecated; the related elements are still supported and will work, but are no longer available in Logi Studio.

Requirements

This topic assumes that SQL Server and SSAS have been installed, configured, and successfully tested.

Your Logi Info application connects to SSAS using the **MSOLAP OLE DB** provider. This provider is *not* shipped with Logi Info but is installed with Microsoft Office, so it may already be on many development machines. For web servers without Office, you'll need to install it.

The ADOMD.NET and MSOLAP providers come as a single package and should be installed on the web server where the Logi application will be running. Which ADOMD.NET provider version you need depends on your SSAS version and it may be included in your MS SQL Server's Feature Pack. This Microsoft documentcan help you identify provider versions and where to find it.

After installing ADOMD.NET, be sure to restart the IIS web server.

In addition, if you're using a Windows client OS version of Windows 7 or earlier, or a server OS version of Windows Server 2008 or earlier, you will need Microsoft XML Core Services (**MSXML**). You can check in Control Panel > Programs to see if it's already installed.

- MSXML 4.0 SP3 is the *minimum* release required and can be downloaded here.
- There are two main components of this installation: the Microsoft XML Parser and the XML SDK. Of these two components, only the Microsoft **XML Parser** needs to be installed for Logi Info and OLAP.

You should also ensure that you have installed the latest SSAS Service Pack for your version of MS SQL Server.

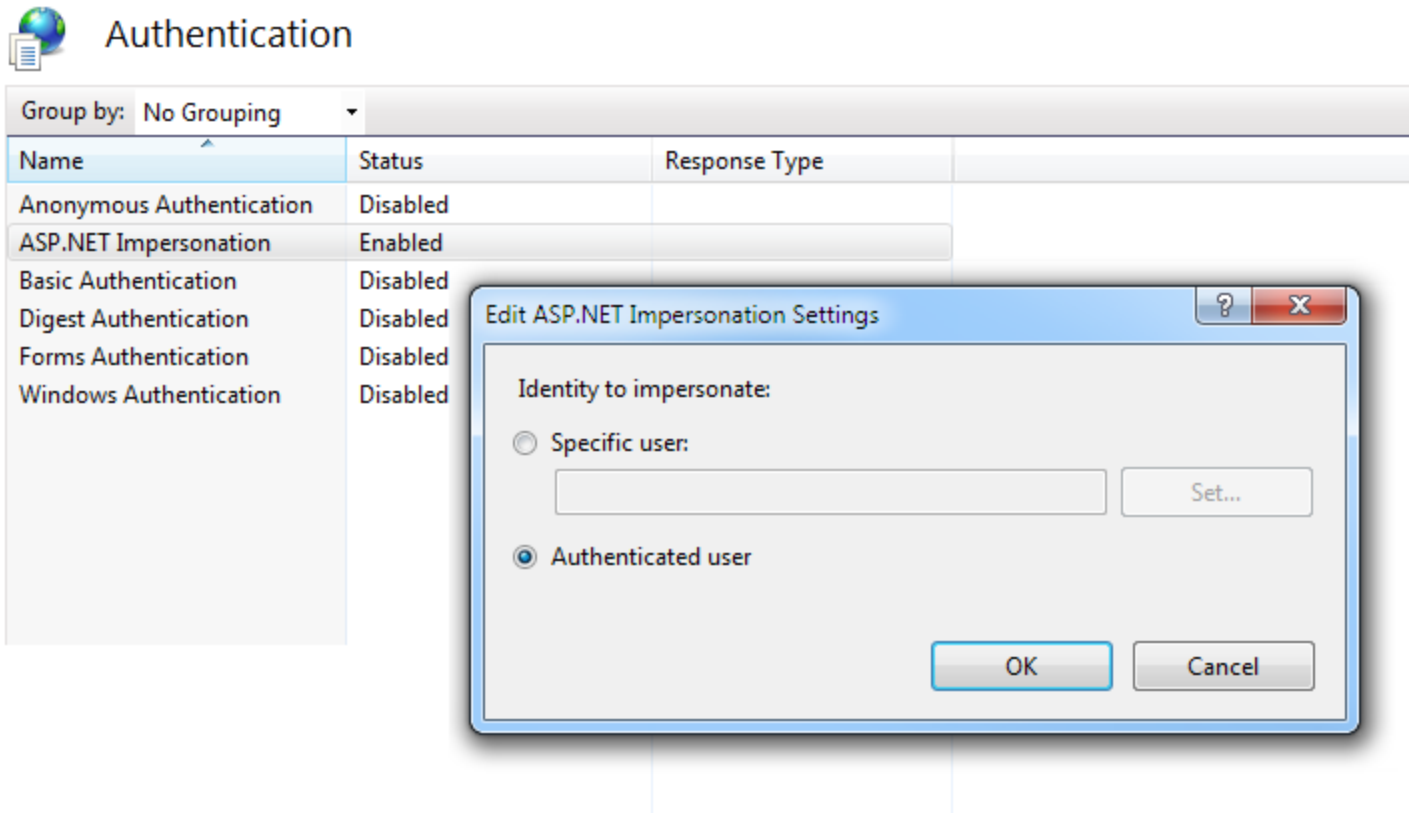
Configuring IIS Authentication

The following examples refer to IIS 7.5 (support ending 2023) but are also applicable to other IIS versions.

Typically, integrated Windows security is used to connect to SSAS and your Logi application's authentication method on IIS must be configured for this. There are different configurations for four possible scenarios, as follows:

A. IIS and SSAS on the Same Server - Use Windows Login Account

If IIS and SSAS are running on the *same* server and you want the Logi application to use the user's Active Directory domain account to access the database. In this case, individual users will need separate logins for the database. To use this approach, configure your IIS authentication for *impersonation*:



This is done in IIS Manager, as shown above, by configuring your application's Authentication feature. In this example, one of the authentication methods available is *ASP.NET Impersonation* - just enable it. Its *Authenticated User* option (shown for completeness here) is the default setting.

If *ASP.NET Impersonation* is not available in your list of methods, you can manually make the change using these steps:

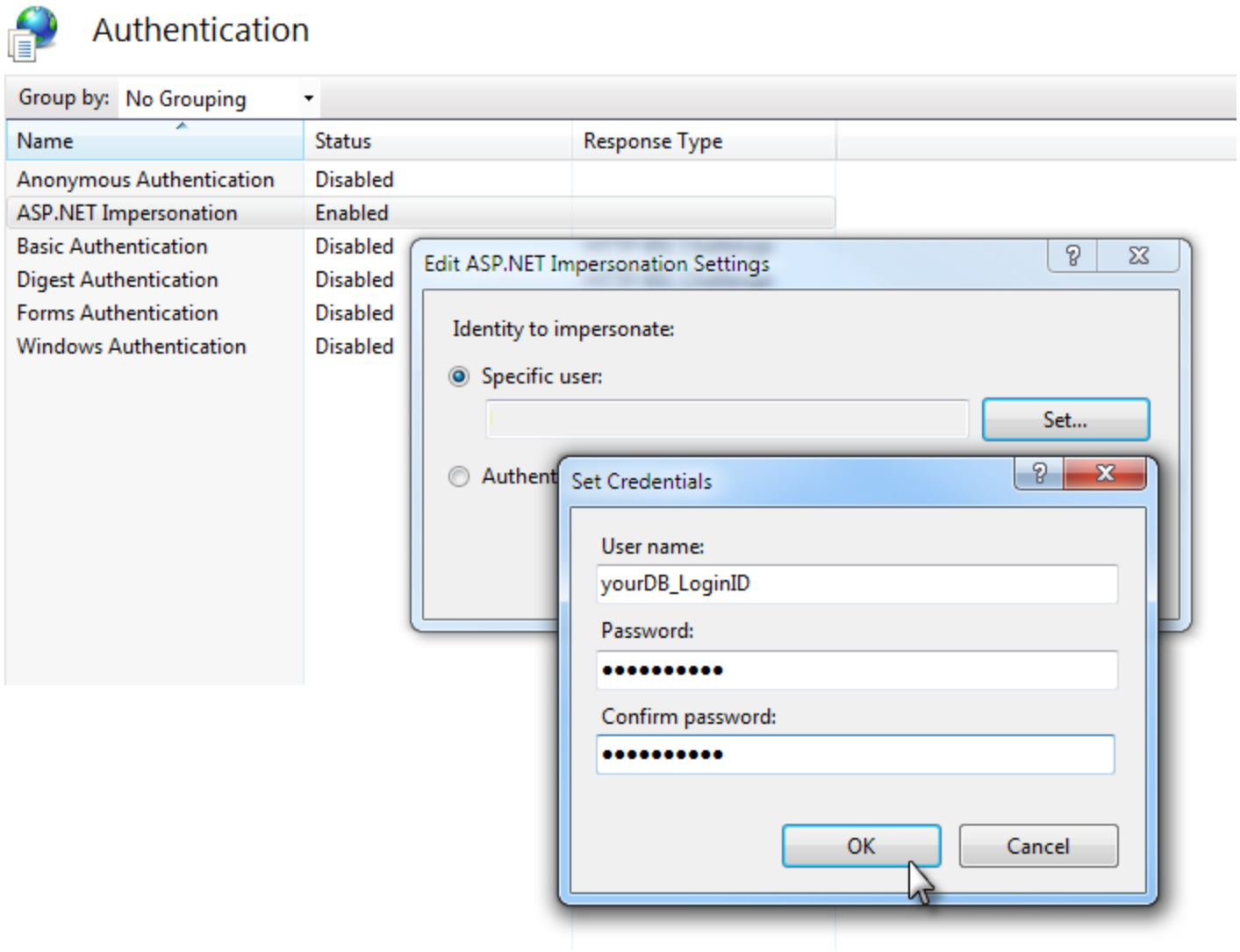
```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
```

```
      <system.web>
<identity impersonate="true"/>

      <!-- DYNAMIC DEBUG COMPILATION
      Set
 compilation debug="true" to insert debugging symbols (.pdb
 information)
      into the
 compiled page. Because this creates a larger file that executes...
```

1. In your Logi application folder, using Notepad or another text editor, edit the file `Web.config`.
2. Add the text highlighted above to the file. The user's Windows account credentials will be passed to SSAS.
3. Save and close the file.

B. Use a Specific Database Login Account

Another approach is to create a single login account for the database, one that all Logi application sessions will use. This presumes appropriate user security will be applied in the application. For this, configure your IIS authentication for *impersonation*:


Authentication
Group by: No Grouping
Name
Status
Response Type
Anonymous Authentication
Disabled
ASP.NET Impersonation
Enabled
Basic Authentication
Disabled
Digest Authentication
Disabled
Forms Authentication
Disabled
Windows Authentication
Disabled
Edit ASP.NET Impersonation Settings
Identity to impersonate:
Specific user:
Set...
Authent
Set Credentials
User name:
yourDB_LoginID
Password:
Confirm password:
OK
Cancel

This is done in IIS Manager, as shown above, by configuring your application's Authentication feature. One of the authentication methods available is *ASP.NET Impersonation* - just enable it - then edit it and select the *Specific User* option. Click **Set...** to supply the database login ID and password you want to use.

When you click **OK**, if the application exists already, the program will attempt a login using the supplied credentials in order to validate them.

If *ASP.NET Impersonation* is not available in your list of methods, you can manually make the change using these steps:

```
      <?xml version="1.0" encoding="UTF-8"?>
      <configuration>
      <system.web>
<identity impersonate="true"
 userName="YourDBLoginID"
 password="yourDBPassword"/>

      <!-- DYNAMIC DEBUG COMPILATION
      Set
 compilation debug="true" to insert debugging symbols (.pdb
 information)
      into the
 compiled page. Because this creates a larger file that executes...
```

1. In your Logi application folder, using Notepad or another text editor, edit the file `Web.config`.
2. Add the text highlighted above to the file. The account credentials entered here will be passed to SSAS.
3. Save and close the file.

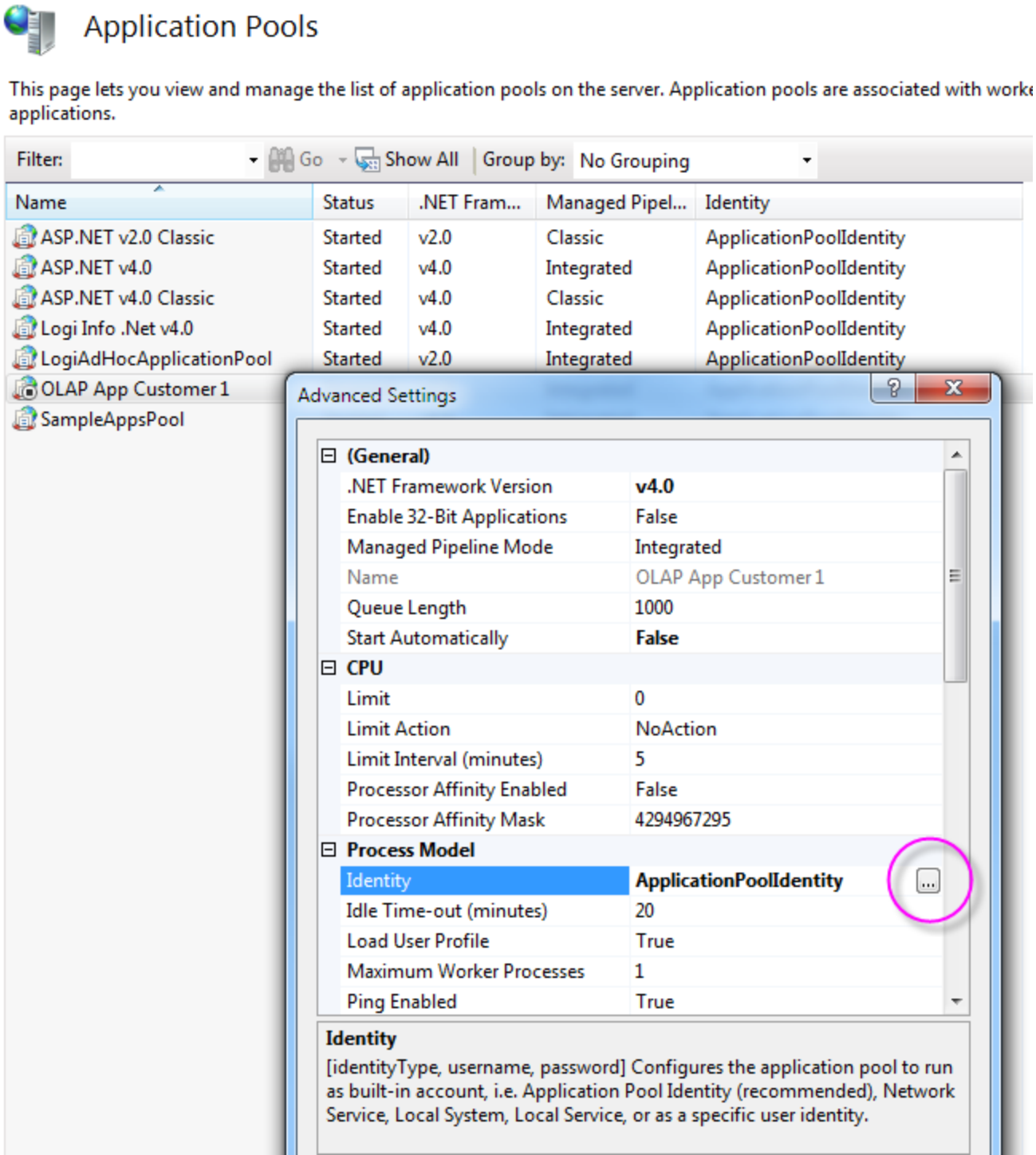
C. IIS and SSAS on the Different Servers - Use Windows Login Account

If IIS and SSAS are running on *different* servers and you want the Logi application to use the user's Active Directory domain account to access the database, then configuration is more complicated. IIS can't directly pass the account information to SSAS - this is known as a dangerous authentication "double-hop" - so you'll need to implement **Kerberos** authentication to help IIS safely pass the account information to SSAS. That procedure is beyond the scope of this document and is discussed in this Microsoft document.

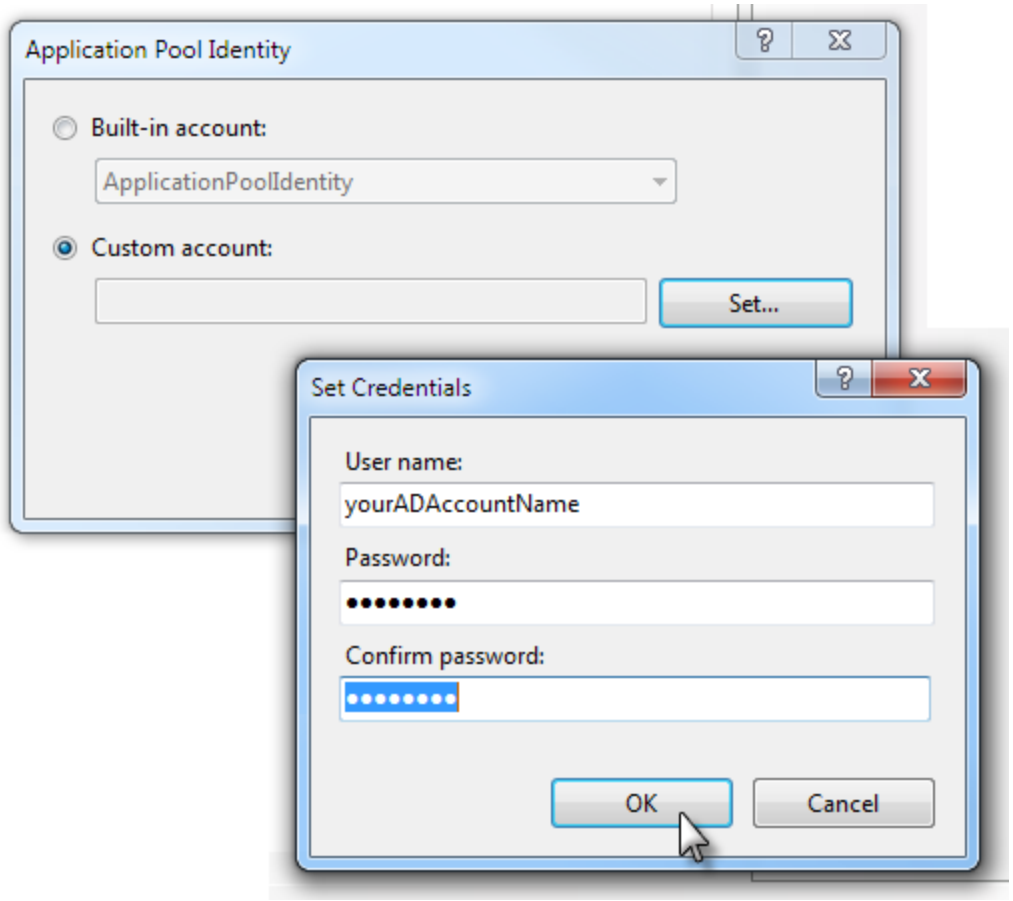
D. Multi-Tenant Configurations

A common "multi-tenant" service approach is to host multiple instances of a Logi application on a single Web server, with separate Application Pools for each instance (among other things, this allows restarting one instance without affecting others). One way to secure access to data while using a single Logi application connection is to specify a unique identity for each Application Pool used. Each identity corresponds to a special Active Directory domain account that has a database login.

Configuring unique Application Pool identities is done in IIS Manager:


Application Pools
This page lets you view and manage the list of application pools on the server. Application pools are associated with worke
applications.
Filter:
Go
Show All
Group by:
No Grouping
Name
Status
.NET Fram...
Managed Pipel...
Identity
ASP.NET v2.0 Classic
Started
v2.0
Classic
ApplicationPoolIdentity
ASP.NET v4.0
Started
v4.0
Integrated
ApplicationPoolIdentity
ASP.NET v4.0 Classic
Started
v4.0
Classic
ApplicationPoolIdentity
Logi Info .Net v4.0
Started
v4.0
Integrated
ApplicationPoolIdentity
LogiAdHocApplicationPool
Started
v2.0
Integrated
ApplicationPoolIdentity
OLAP App Customer 1
SampleAppsPool
Advanced Settings
(General)
.NET Framework Version
v4.0
Enable 32-Bit Applications
False
Managed Pipeline Mode
Integrated
Name
OLAP App Customer 1
Queue Length
1000
Start Automatically
False
CPU
Limit
0
Limit Action
NoAction
Limit Interval (minutes)
5
Processor Affinity Enabled
False
Processor Affinity Mask
4294967295
Process Model
Identity
ApplicationPoolIdentity
Idle Time-out (minutes)
20
Load User Profile
True
Maximum Worker Processes
1
Ping Enabled
True
Identity
[identityType, username, password] Configures the application pool to run as built-in account, i.e. Application Pool Identity (recommended), Network Service, Local System, Local Service, or as a specific user identity.

1. Open the Application Pools node in the Connections panel.
2. Select the application pool you want to configure and select its *Advanced Settings*.
3. Select the *Identity* property, as shown above, and click its browse button.



4. In the Application Pool Identity dialog box that appears, check *Custom Account* and click **Set...**
5. Enter the Active Directly domain account name and password to be used to access SSAS.

6. Click **OK** as needed to exit.

The Microsoft document *Specifying an Application Pool Identity (IIS7)*provides more information about this process.

Which Configuration Should You Use?

Let's review the basic security terminology: *authentication* identifies the user as valid, and *authorization* determines which resources the authenticated user can access. In production environments, it's not likely that IIS and SSAS will be running on the same system, so scenarios B, C, and D above are the most likely authentication options.

The most common approach used by Logi developers is B, which locates the handling of authorization *in the Logi application*, using Logi Security to control which reports and activities, and therefore data, users can access. Using one specific login to access the database is suitable for this and minimizes account maintenance headaches. This is widely used by Logi customers for both OLAP and regular database access. Each user has their own session on the web server and that translates into their own connection to the database, so connection-related performance constraints are not generally an issue.

Approaches C and D, which locate authorization *in the database* itself, essentially recreate the accounts and/or roles that appear in the application authentication mechanism in the database as well. This requires additional administrative overhead (for example, accounts must be maintained in *both* Active Directory and in SQL Server) and connection complexity. Approach D is sometimes used in multi-tenant database scenarios, but it's not a required approach. Developers who choose to use these approaches will need extensive expertise with Windows, IIS, and SSAS security configuration techniques in order to succeed.

Configuring the Connection Element

To connect to SSAS, a **Connection.OLAP** element needs to be added in your Logi application's _Settings definition and its **ADOMD Connection String** attribute needs to be configured.

Connection Strings can include a variety of properties and the examples provided below are functional but not all-inclusive. More information about SSAS connection string properties can be found in this Microsoft document.

If you have more then one MSOLAP provider installed you may need to specify the version of the provider in connection strings:

Provider	SSAS Version
MSOLAP.3	2005
MSOLAP.4	2008
MSOLAP.5	2012
MSOLAP.6	2014, 2016

Just specifying *MSOLAP* as the provider causes the *latest* version of MSOLAP installed on the system to be used.

SSAS 2012, 2014, 2016

A typical value for the ADOMD Connection String attribute for SSAS 2012, 2014, and 2016 is:

```
Provider=MSOLAP.5;Integrated Security=SSPI;Persist Security
 Info=True;Initial Catalog=Adventure Works DW 2008R2;Data
 Source=AW-SRV01;MDX Compatibility=1;Safety Options=2;MDX Missing Member
 Mode=Error
```

SSAS 2008

A typical value for the ADOMD Connection String attribute for SSAS 2008 is:

```
Provider=MSOLAP.4;Data
 Source=IPaddress\instancename;IntegratedSecurity=SSPI;Initial
 Catalog= yourDatabasename;Persist Security Info=True
```

In the example above, the configuration of member "uniqueness" is not done in the connection string. Instead, it's accomplished by using the Business Intelligence Development Studio (BIDS) tool, by setting these Cube Dimension properties:

```
      HierarchyUniqueNameStyle = IncludeDimensionName
      MemberUniqueNameStyle = NamePath
```

See your SSAS 2008 documentation for more information about the use of BIDS.

SSAS 2005

A typical value for the ADOMD Connection String attribute for SSAS 2005 is:

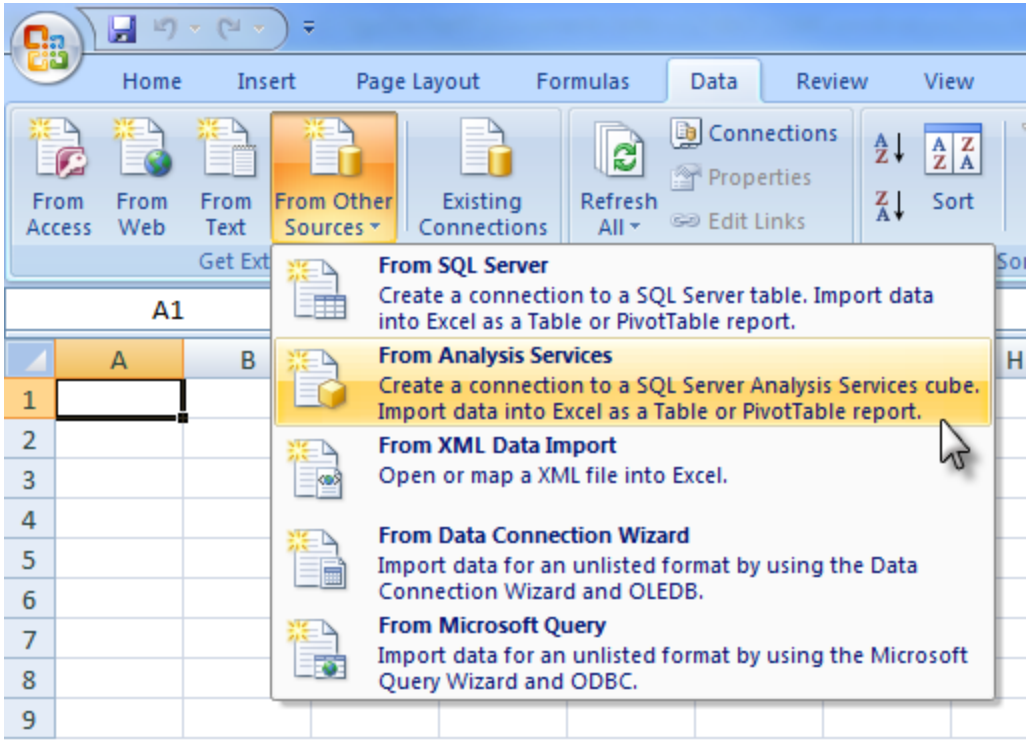
```
Provider=MSOLAP.3;Data
 Source=yourServername;IntegratedSecurity=SSPI;Initial
 Catalog=yourDatabasename;Client Cache Size=25;Auto Synch
 Period=10000;MDX Unique Name Style=2;Default MDX Visual Mode=1;
 Persist Security Info=False
```

If you're using multiple SSAS 2005 instances, a typical value is:

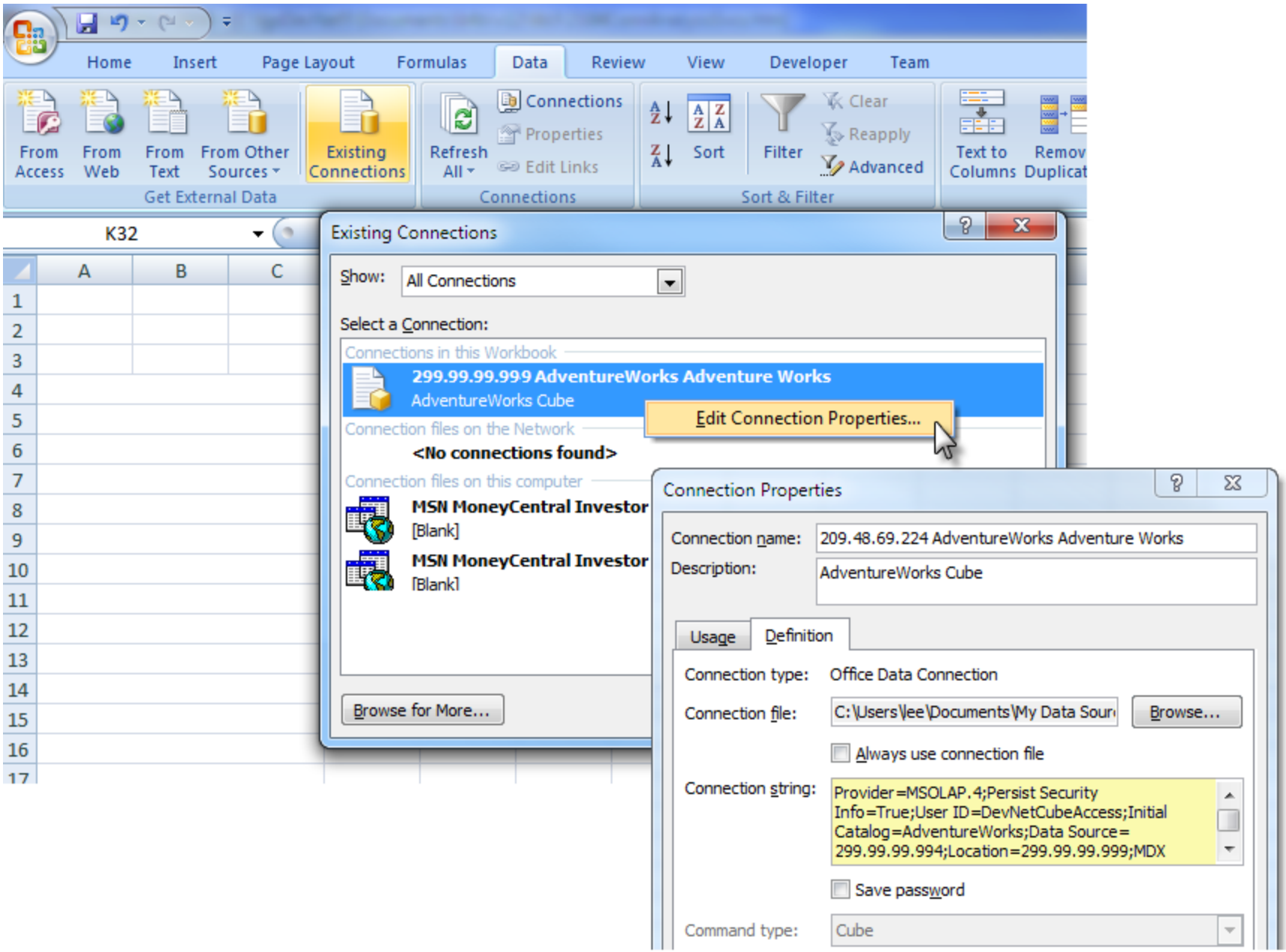
```
      Provider=MSOLAP.3;Data
 Source=IPaddress\instancename;IntegratedSecurity=SSPI;Initial
      Catalog=yourDatabasename;Client Cache Size=25;Auto Synch
 Period=10000;MDX Unique Name Style=2;Default MDX Visual Mode=1; Persist
 Security Info=False
```

Using Excel to Create Connection Strings

Microsoft Excel is capable of connecting to SSAS and you can use it to help you create a proper connection string:



First, use Excel's Data Connection wizard in the Data tab, shown above, to create a connection to SSAS. It will generate the proper connection string with minimal input from you. Once you complete the wizard's steps, you can view the new connection's properties:


Home
Insert
Page Layout
Formulas
Data
Review
View
Developer
Team
From Access
From Web
From Text
From Other Sources
Existing Connections
Get External Data
Refresh All
Connections
Properties
Edit Links
Connections
Sort
Filter
Clear
Reapply
Advanced
Sort & Filter
Text to Columns
Remov Duplicat
K32
Existing Connections
Show:
All Connections
Select a Connection:
Connections in this Workbook
299.99.99.999 AdventureWorks Adventure Works
AdventureWorks Cube
Edit Connection Properties...
Connection files on the Network
<No connections found>
Connection files on this computer
MSN MoneyCentral Investor
[Blank]
MSN MoneyCentral Investor
[Blank]
Browse for More...
Connection Properties
Connection name:
209.48.69.224 AdventureWorks Adventure Works
Description:
AdventureWorks Cube
Usage
Definition
Connection type:
Office Data Connection
Connection file:
C:\Users\lee\Documents\My Data Sour
Browse...
Always use connection file
Connection string:
Provider=MSOLAP.4;Persist Security Info=True;User ID=DevNetCubeAccess;Initial Catalog=AdventureWorks;Data Source=299.99.99.994;Location=299.99.99.999;MDX
Save password
Command type:
Cube

In the Data tab, view the existing connections, right-click the connection you just added, and view its properties, as shown above. In the Properties Definition tab, you'll see the connection string the wizard created. You can copy this (being sure you select and copy *all* of it - you may need to scroll the text box) and paste it into your Connection element in the _Settings definition.

Possible Firewall Issues

Be aware that SSAS uses a different port than SQL Server itself for communications. If you're having trouble connecting to SSAS, ensure that your firewall, if one's in use, is configured properly to allow SSAS access. This Microsoft documentprovides configuration details for Windows Firewall.

Dataviews

Logi Data Services technology uses Dataviews to define datasource connections, queries, and data enrichment. This topic discusses Dataviews and the tools for creating and managing them.

The following sections are covered in this topic:

- About Logi Platform Services Technology
- Related Logi Info Elements

To use this technology, you must install the **Discovery Module 3.0+** add-on.

About Logi Services Technology

The elements discussed here are available in Logi Info v12.5 but have been deprecated in later Info versions; consult the Release Notes for specific details. **Logi Services** technology has been introduced into Logi Info, through the Discovery add-on module. It's Logi Data Service provides you with an additional, advanced means of data retrieval based on the "Dataview", a definition that specifies data connection information, query details, and data enrichment details.

That definition is stored in a system database, is called using special DataLayer elements, and can be altered at runtime using special Data Service ("Ds") elements (see the next section). When executed at the server, the Dataview retrieves the data, enriches and paginates it, and makes it available. Some of the benefits of this include:

- Decoupling of developer and data architect duties
- Reusable Dataviews, available across multiple applications
- Shared access to stored Dataviews by multiple users
- Application of security at the data level

- Better support for Self-Service analytics
- Excellent performance for very large datasets (250M+ rows)

When used with Logi Info's **DataLayer.Data Services** element, after the Dataview runs, the data is converted into the standard Logi XML format and made available to other elements for visualizations, tables, etc. When used with the **DataLayer.Data Services** element, the data is made directly available to the Thinkspace.

Dataviews can be created using the Dataview Authoring tool accessible from Logi Studio and they can also be created using the Logi Platform Services REST API. Detailed instructions for these activities are presented in *Dataview Authoring*.

The Dataviews used with Logi Info can retrieve data from a variety of databases:

- AWS Redshift
- Infobright
- JDBC database providers
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Snowflake
- Vertica

If our **DataHub 3.0+** product is installed, then Dataviews can *also* retrieve and join file-, cloud-, and application-based data, and cache it in a separate data store. Data sources include:

- Amazon Dynamo DB
- Eloqua

- Facebook
- Financial Accounts (OFX)
- Google Analytics
- Marketo
- Microsoft Dynamics CRM On Premise
- Microsoft Dynamics CRM Online
- Microsoft Dynamics GP
- Microsoft Excel and CSV files
- Netsuite
- OData
- Quickbooks On Premise
- Quickbooks Online
- Quickbooks POS
- Salesforce
- SAP Netweaver
- Twitter

Related Logi Info Elements

The elements discussed here are available in Logi Info v12.5 but have been deprecated in later Info versions; consult the Release Notes for specific details. The following Logi Info elements can be used with Logi Data Services:

Element	Description
Connection.Logi Application Services	The Connection element that connects to the Logi Application Services web service installed with the Discovery Module. For more information, see "Datasource Connections" on page 10.

Element	Description
DataLayer.Data Services	The DataLayer element that works with the Connection.Logi Application Services element to retrieve data using Dataviews.
The DsAggregate Column	This element is used with DataLayer.Data Services to add a new column to the data containing an aggregated value of the data in one of the other columns.
The DsCalculated Column	This element is used with DataLayer.Data Services to add a new column to a data containing a value that's the result of a expression. The syntax for creating calculated values is available in the *Dsexpression Reference*.
The DsCondition Filter	This element is used with DataLayer.Data Services to filter out data rows. It's analogous to using a *WHERE* clause in a SQL query and adds a filtering operation to a Dataview. The syntax for filtering values is available in the *Dsexpression Reference*.
DsGroup	This element is used with DataLayer.Data Services to add a grouping operation to Dataview processing. It also gives you the ability to create aggregate values from the grouped rows.
DsSort	This element is used with DataLayer.Data Services to add a sortingoperation to Dataview processing.
SuperWidget	This element allows you to add widgets, created using the Dataview Authoring and SuperWidget Authoring tools, into Logi Info applications. For more information, see *SuperWidgets*.

Data Definitions

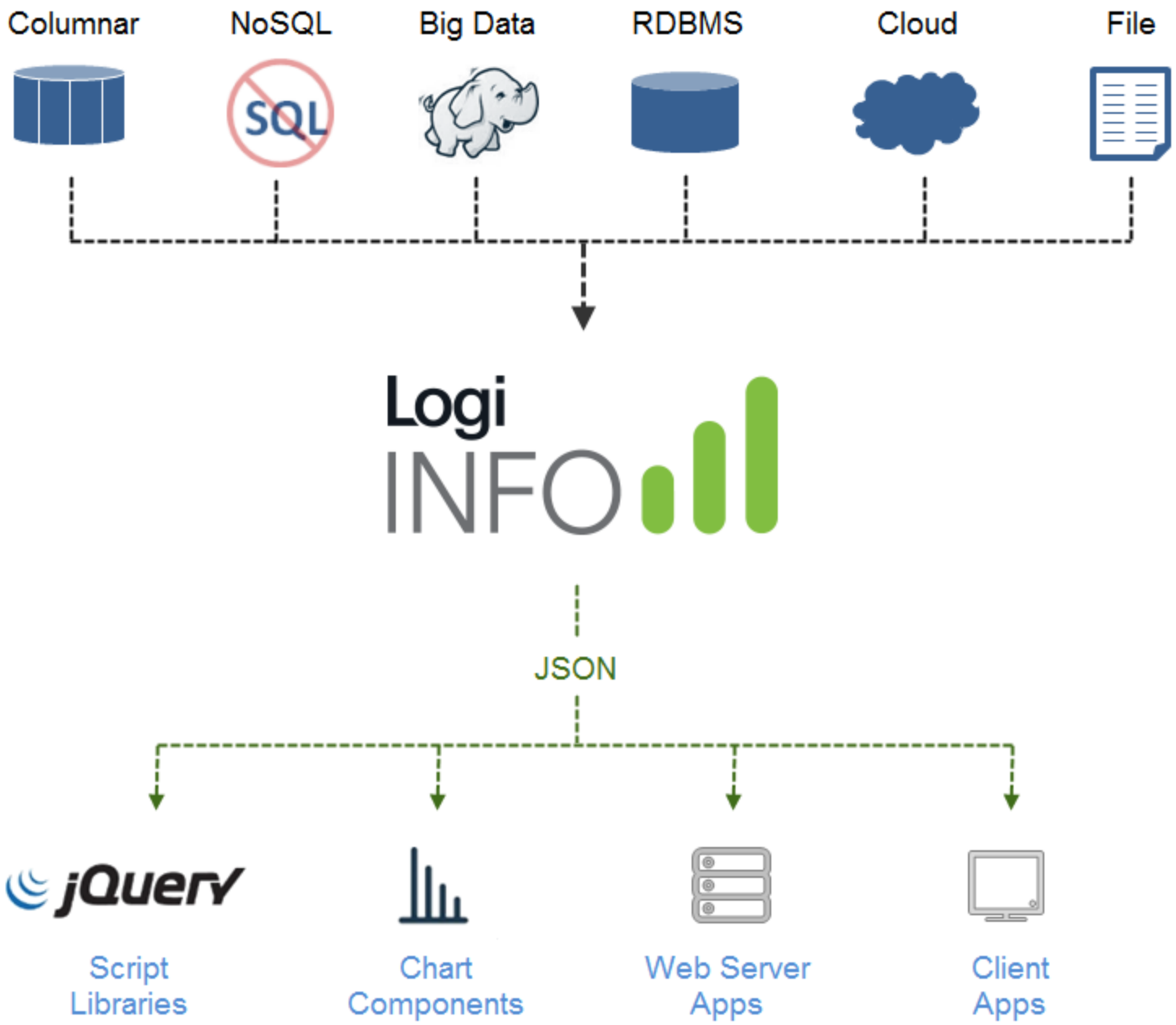
Data Definitions allow developers to leverage Logi Info's ability to connect to a variety of datasources in order to create a powerful JSON and XML data provider for Logi applications, non-Logi applications, and client browsers.

The following topics introduce the data definitions used to retrieve data:

About Data Definitions

Logi Info applications are capable of connecting to, and retrieving data from, a wide range of datasources. They also allow retrieval from multiple datasources simultaneously and can provide SQL-like JOINs between non-SQL datasets.

Data definitions allow the developer to retrieve data from these sources, manipulate it, and then make it available as a JSON data stream to a variety of data consumers.


Columnar
NoSQL
Big Data
RDBMS
Cloud
File
SQL
Logi
INFO
JSON
jQuery
Script Libraries
Chart Components
Web Server Apps
Client Apps

The diagram shown above illustrates this concept. Creating a Logi data definition is simple and uses the same elements and techniques used in Logi report definitions. Writing code that consumes the JSON data is also easy and a variety of examples are provided in the later sections of this topic.

The JSON data produced can be accessed by Logi apps, directly from browsers, and by any application written in a programming language that supports HTTP requests.

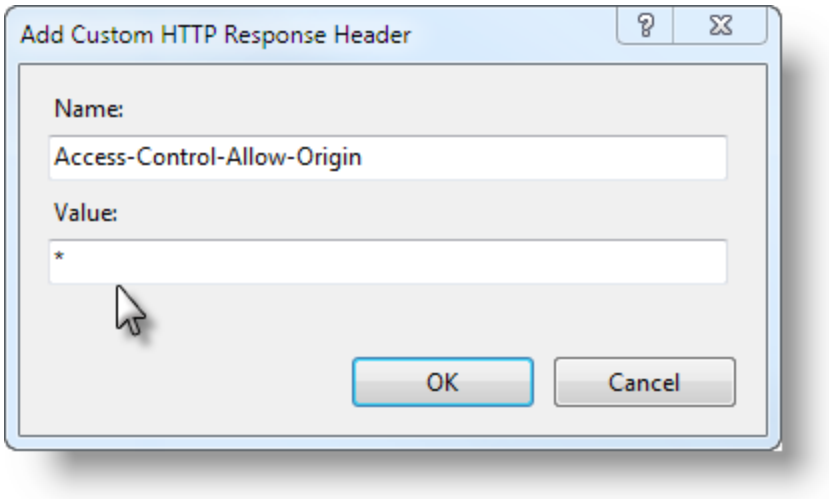
Data definitions can also create both JSON and XML data.

Browsers and Cross-Origin Resource Sharing

When the client accessing the Logi data definition is a browser that's *not* on the same domain, you'll need to use the **Cross-Origin Resource Sharing** (CORS) mechanism to guarantee access. The CORS standard describes HTTP headers which provide browsers with a way to request remote URLs when they have permission. CORS is supported by most modern browsers.

CORS is *not* necessary when *web server applications* are consuming the data.

CORS is implemented for Logi data definitions by configuring your web server's HTTP Response Header.



The example above shows this being done for IIS, using the IIS Manager application. It can also be done by modifying the `web.config` file directly.

```
<filter>
<filter-name>CorsFilter</filter-name>
<filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
</filter>
<filter-mapping>
```

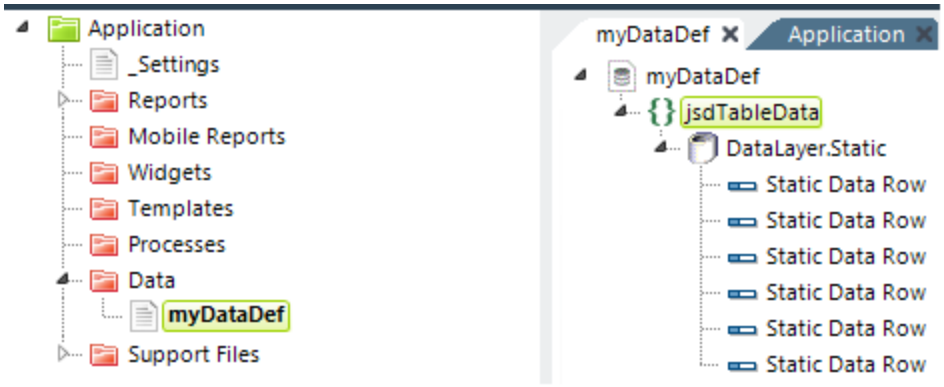
```
<filter-name>CorsFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

For Apache Tomcat (starting with v7.0.41), the code shown above can be added to `$CATALINA_BASE/conf/web.xml` or `WEB-INF/web.xml`. Other Linux-UNIX web servers can be configured similarly. Refer to your server's documentation.

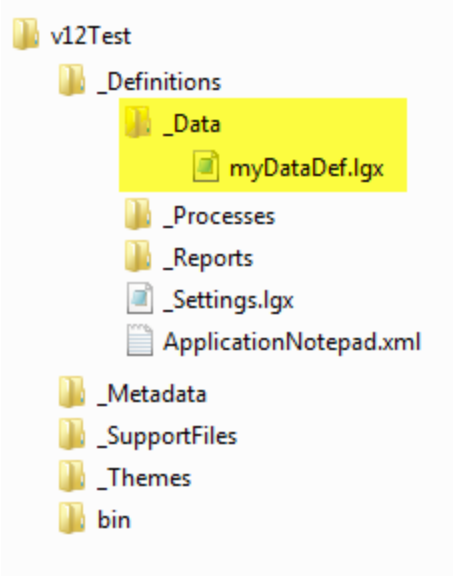
This external site, Enable CORS, provides a lot of useful information about CORS and has server configuration examples.

Data Definition Basics

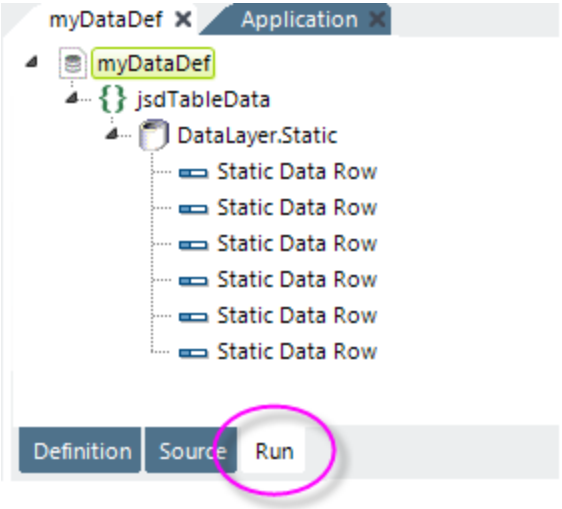
Data definitions are very similar to report and process definitions.

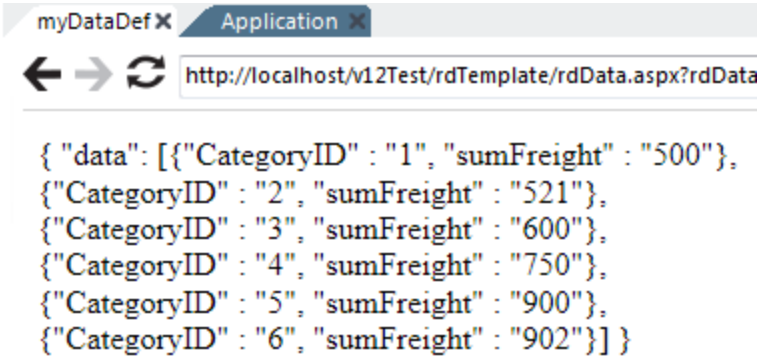


In Logi Studio, they're organized beneath the **Data** folder in the Application panel, as shown above.



In the file system, they're stored in the _Data folder, which is in the _Definitions folder along with the _Processes and _Reports folders, as shown above.



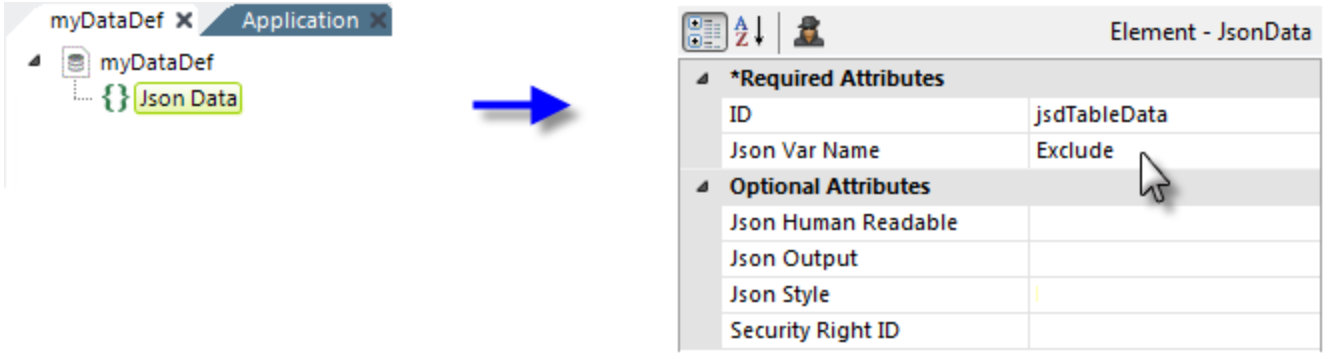


You can test a Data definition in Studio using the **Run** tab at the bottom of the Workspace panel, shown above left. When you do, the actual JSON data generated will be displayed, as shown above right.

As with any other definition, you can use **Debugger** links to see the details of the definition execution.

Creating a JSON Data Definition

A JSON Data definition uses a small subset of the standard Logi Info elements, most of them related to data retrieval and manipulation.

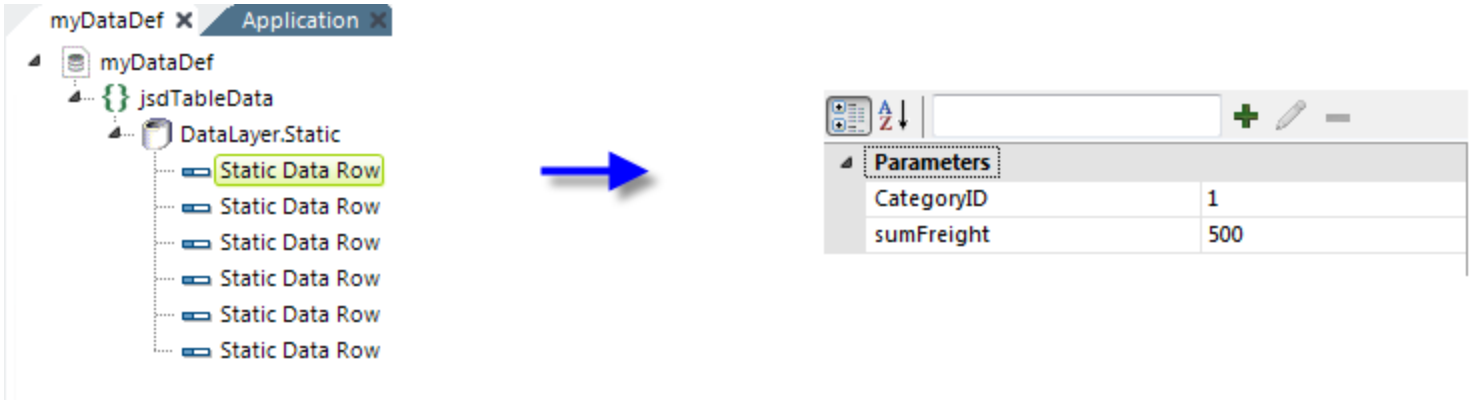


The first element that needs to be added in any JSON Data definition is a **Json Data** element,asshownabove. It will run a datalayer, insert the results into a JSON JavaScript variable, and streams it.

Its attributes are:

Attribute	Description
ID	(Required) A unique element ID.
Json Var Name	(Required) When used in Data definitions, this should always be set to *Exclude*.
Json	Specifies whether spaces and LFs are used to format the output JSON data, for easier debugging.

Attribute	Description
Human Readable	
Json Output	When used in Data definitions, this should always be blank, or *Inline.*
Json Style	This attribute specifies the format of the data output; either as *RowsToObjects,RowsToValueArrays,* or as *ColumnsToPropertyArrays*. See the discussion below with output examples. Default: *RowsToObjects*
Security Right ID	Controls access to this element when using Logi Security. Provide the ID of a Right defined in the application's _Settings definition and only users that have a Role referenced in the Right will be able to use the data. Multiple Right IDs, separated by commas, may be entered.



Next, we add a datalayer beneath the Json Data element. In the example above, we've used DataLayer.Static to make it easy to see the data. And we're done. You can, of course, have more complex data arrangements, using any of the usual datalayer types, filters, JOINs, and other elements to retrieve and manipulate the data set.

Using Different Json Styles

The Json Data element can format its data in three different ways, using its **Json Style** attribute:

```
{ "data": [{"CategoryID" : "1", "sumFreight" : "500"}, {"CategoryID" : "2", "sumFreight" : "521"},
{"CategoryID" : "3", "sumFreight" : "600"}, {"CategoryID" : "4", "sumFreight" : "750"}, {"CategoryID"
: "5", "sumFreight" : "900"}, {"CategoryID" : "6", "sumFreight"
: "902"}] }
```

If Json Style is left blank or set to *RowsToObjects*, the data from our example will be formatted as shown above. The datalayer rows are serialized into an array of objects, where each object represents a data row and the objects have a property for each column.

```
{ "data": { "CategoryID" : ["1", "2", "3", "4", "5", "6"], "sumFreight" : ["500", "521", "600", "750",
"900", "902"] } }
```

If Json Style is set to *ColumnsToPropertyArrays*, the data from our example will be formatted as shown above. The datalayer is serialized into a single object, where each column is a property with all the row values contained in an array.

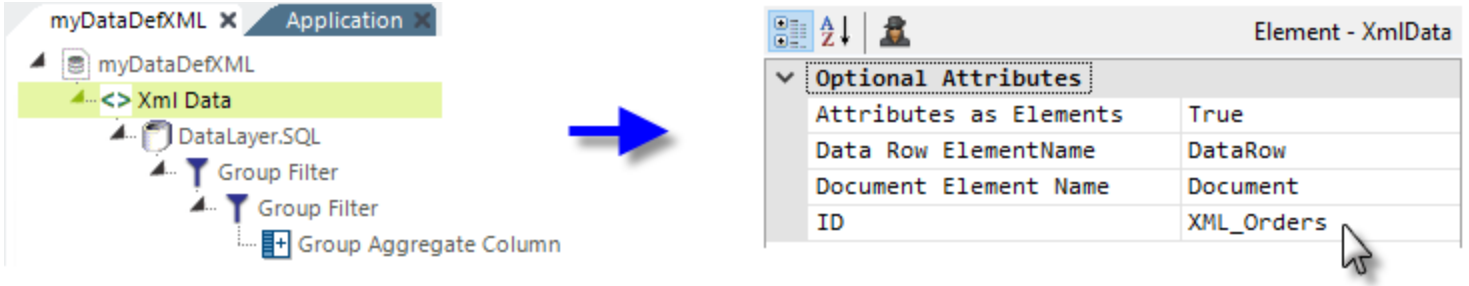
```
{ "data": [["1", "500"], ["2", "521"], ["3", "600"], ["4", "750"], ["5", "900"], ["6", "902"]] }
```

Or, finally, if it's set to *RowsToValueArrays*, the data will be formatted as shown above. This serializes the data into a single array with a child array for each row in the data. Each value in the array represents a column in the data.

You need to select the format that works with the client applications that will consume the data.

Creating an XML Data Definition

An XML Data definition uses a small subset of the standard Logi Info elements, most of them related to data retrieval and manipulation.



The first element that needs to be added in any XML Data definition is an **XML Data** element. It runs a datalayer, inserts the results into an XML document, and streams it.

Its optional attributes are:

Attribute	Description
Attributes as Elements	Specifies the format of the data values. The values are normally expressed as attributes in XML, but may be represented instead as elements with text nodes. When this attribute is set to *True*, values are contained in elements. For example: If *False* (the default) is specified, the output will look like: `<Root>`

Attribute	Description
	<Order OrderID="12345" Freight="1.12" /> <Root> If *True* is specified, the output will look like: <Root> <Order> <OrderID>12345</ID <Freight>1.12</Freight> </Order> <Root>
Data Row ElementName	Specifies the name for XML elements which represent each of the data rows. The default is the Xml Data element's ID attribute.
Document Element Name	Specifies the name of the output XML's root element. The default is the Xml Data element's ID attribute.
ID	A unique element ID.

Retrieving Data

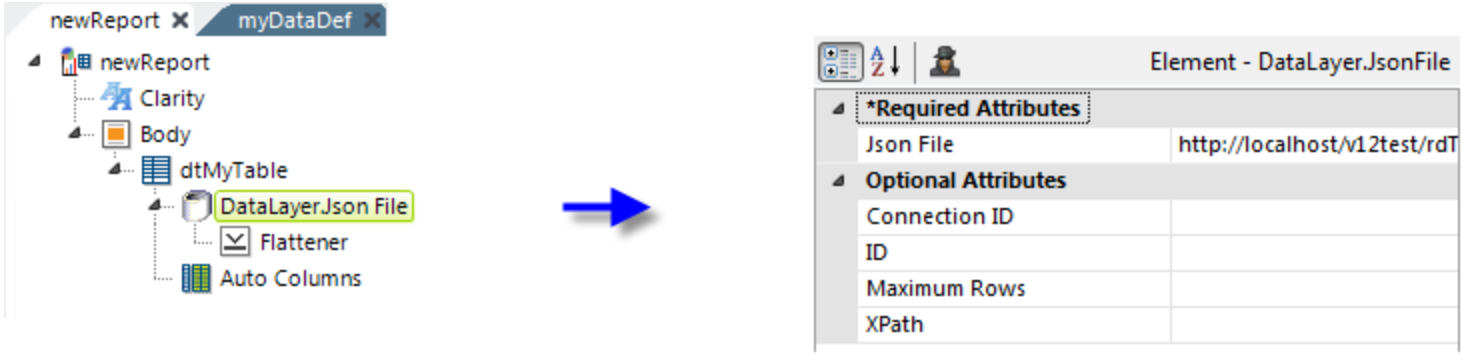
Retrieving the Logi data stream is extremely easy, simply use a URL in this format:

```
https://yourWebServer/yourLogiApp/rdTemplate/rdData.aspx?rdData=yourDataDefinition
```

In the following sections, you'll see examples of data retrieval, from within several Logi applications. We'll also see how additional Query String parameters in the URL can be used to dynamically condition the data.

Retrieving Data Example: Getting Data for a Logi Application

Here's a very simple example of a Logi application that uses the data from the Create a Data Definition topic. The Json Style is blank (*RowsToObjects)* in that example.



In the report definition above, Data Table and DataLayer.Json File elements have been added. The datalayer's Json File attribute has been set to:

https://localhost/v12test/rdTemplate/rdData.aspx?rdData=myDataDef

A Flattener element has been used to flatten the JSON data array into rows and columns for use in the Data Table and an Auto Columns element handles our columns. For more information on these elements, see *The Flattener* and *Auto Columns*.

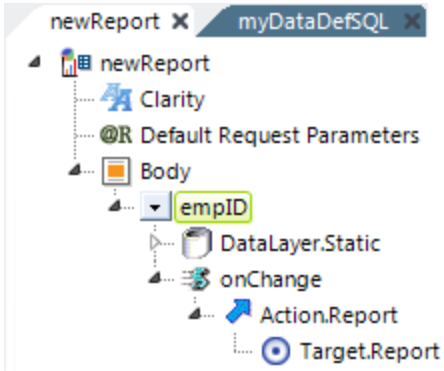
CategoryID	sumFreight
1	500
2	521
3	600
4	750
5	900
6	902

And, if we preview the report definition, the resulting output is shown above.

Retrieving Data Example: Using Parameters for Dynamic Data

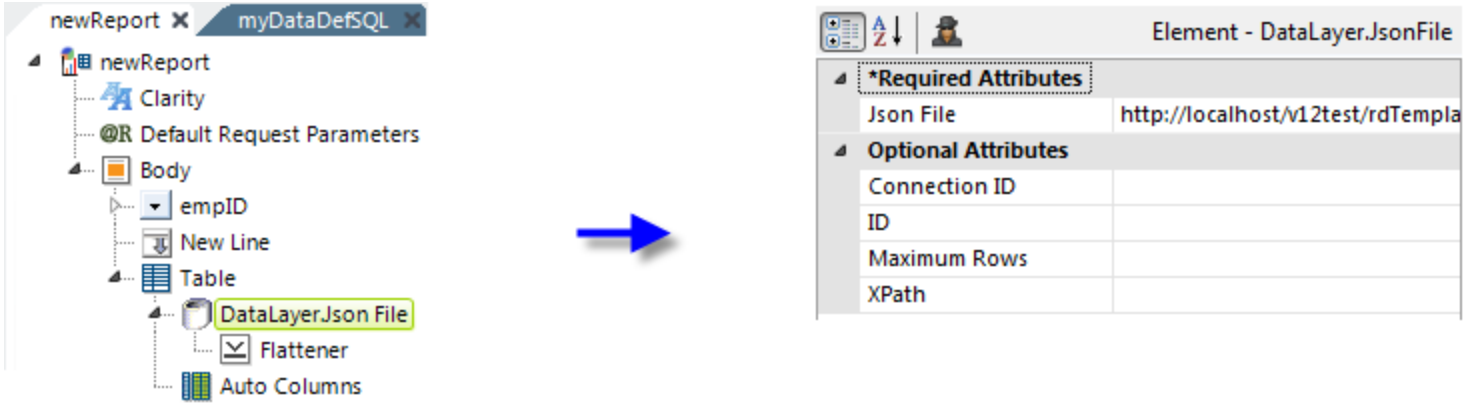
This example is similar to the previous one, but introduces the ability to condition the data using Query String parameters.

A new data definition, shown above, includes both a **Default Request Parameters** element and a **DataLayer.SQL** element. The Default Request Params ensures that the SQL query, which uses a Request token, will always have a valid WHERE clause value.



We'll start our example report definition, shown above, by adding an **Input Select List**, with a datalayer and event handler, to allow the user to select an Employee ID. A Default Request Params element will ensure that the select list always has a default

value and the event handler just refreshes the report (and updates the select list Request variable) when a new Employee ID selection is made.



We'll finish our report definition with the same set of elements we used previously: Data Table, DataLayer.Json File, Flattener, and Auto Columns. The datalayer's Json File attribute is set to:

```
https://localhost/v12test/rdTemplate/rdData.aspx?rdData=myDataDefSQL&inpEmployeeID=@Request.empID~
```

Note the use of an extra Query String parameter, with a value set from a Request token.

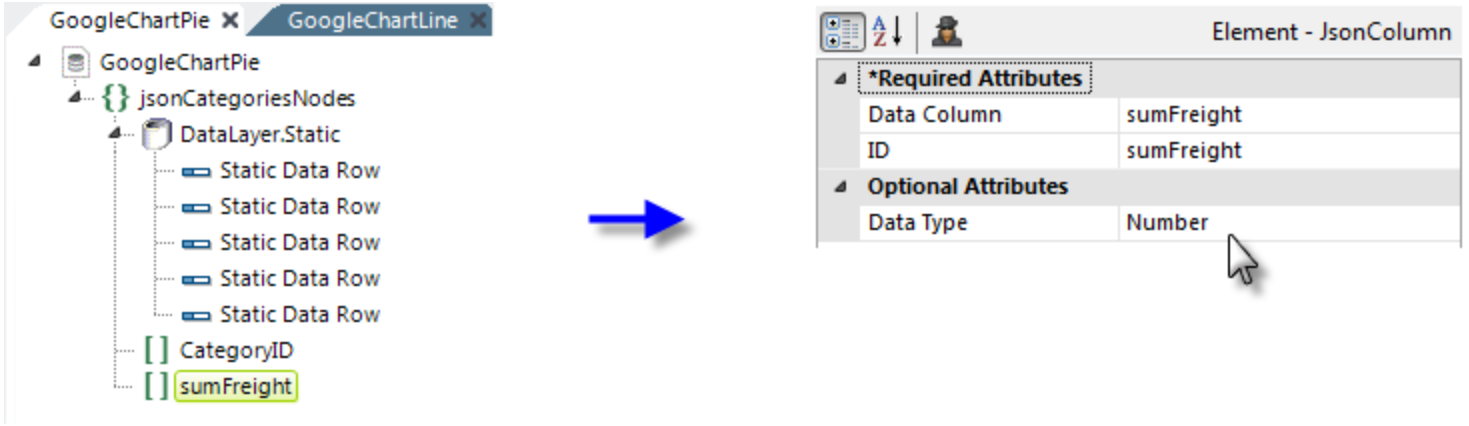
Select Employeed ID: 1

OrderID	CustomerID	EmployeeID	OrderDate
10258	ERNSH	1	7/17/1996 12:00:00 AM
10270	WARTH	1	8/1/1996 12:00:00 AM
10275	MAGAA	1	8/7/1996 12:00:00 AM
10285	QUICK	1	8/20/1996 12:00:00 AM

And the resulting report looks like this when previewed. Changing the Employee ID selection causes the data to refresh.

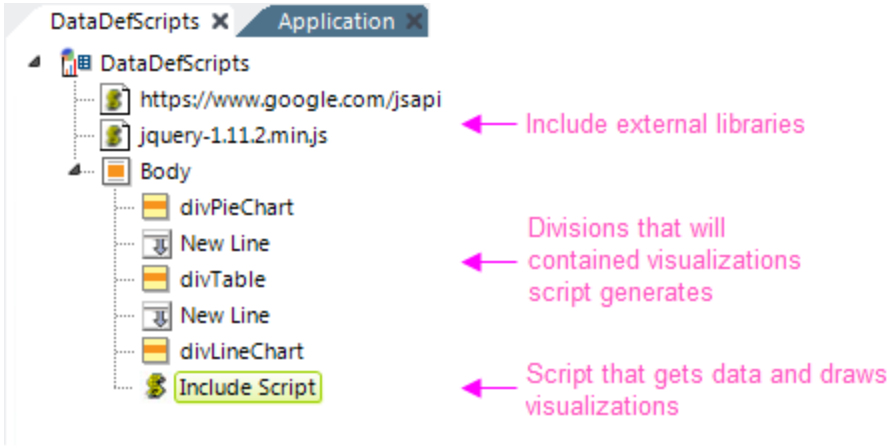
Retrieving Data Example: Using 3rd-Party APIs

Now let's look at a somewhat more complicated example that uses 3rd-party APIs (JQuery and Google) to draw three visualizations, using data from *two* Data definitions. The Data definitions are very similar, except one has more data points.



The Data definitions shown above are much like those we've seen earlier, except for two differences. The Json Data element's **Json Style** attribute is set to *RowsToValueArrays* and two **Json Column** elements are used.

By default, the Json Data element generates JSON data with the same column names as those output by the datalayer and all the JSON data types are strings. The **Json Column** element allows you to specify which columns will be included in the data, their variables names, and their data types. The Data Column attribute is the column name in the datalayer, the ID is column name in the JSON data, and the Data Type can be *Number*, *String*, or *Date*.



In our report definition, shown above, we've added two **Include Script File** elements to include the scripting libraries (one local, one external) we want to use. Empty **Division** elements, with real <DIV> tag output, are provided as the containers of the charts and Data Table we'll draw.

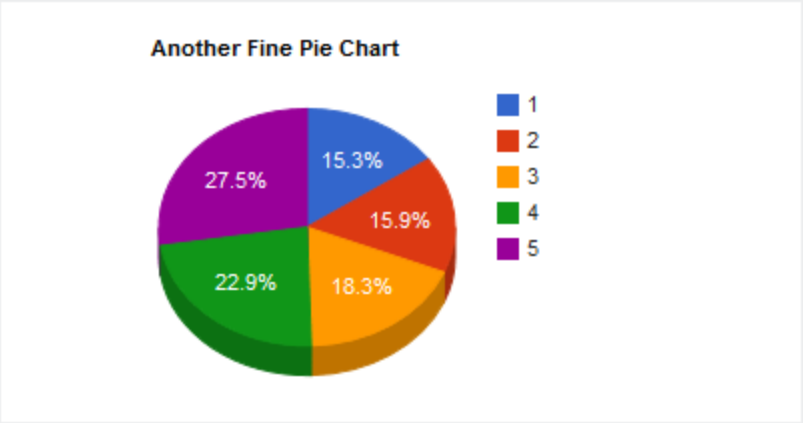
Finally, an **Include Script** element contains our JavaScript, which looks like this:

```
// load the Visualization API and the piechart package
google.load('visualization', '1', {'packages':['corechart']});
google.load("visualization", "1", {packages:["table"]});

// set a callback to run when the Google Visualization API is loaded
google.setOnLoadCallback(drawVisuals);

function drawVisuals() {

// get data from the first data definition
```

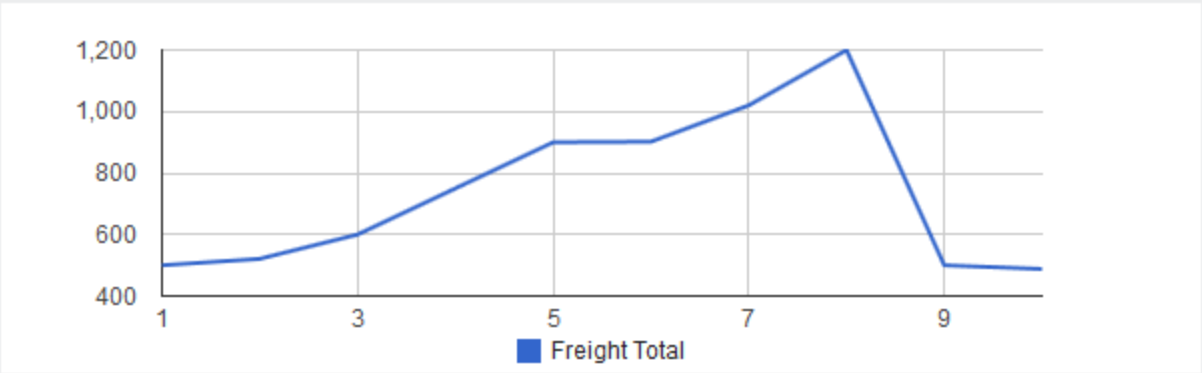
```
var jsonData = $.ajax({
url: "http://localhost/v12test/rdTemplate/rdData.aspx?rdData=GoogleChartLine",
dataType:"json",
async: false
}).responseJSON;

// create a table of the first data set
var data = new google.visualization.DataTable();
data.addColumn('number', 'Category ID');
data.addColumn('number', 'Freight Total');

// apply the data
data.addRows(jsonData.data);

// get datafrom a second data definition
var jsonData2 = $.ajax({
url: "http://localhost/v12test/rdTemplate/rdData.aspx?rdData=GoogleChartPie",
dataType:"json",
async: false
}).responseJSON;

// create a table of the second data set
var data2 = new google.visualization.DataTable();
data2.addColumn('string', 'Category ID');
data2.addColumn('number', 'Freight Total');
```

```
data2.addRows(jsonData2.data);

// instantiate and draw the Data Table, with some options
var table = new google.visualization.Table(document.getElementById('divTable'));
table.draw(data, {showRowNumber: false});

// instantiate and draw the line chart, using the same data as the table
var lineChart = new google.visualization.LineChart(document.getElementById('divLineChart'));
lineChart.draw(data, {width: 600, legend: 'bottom'});

// instantiate and draw the pie chart, with some options
var pieChart = new google.visualization.PieChart(document.getElementById('divPieChart'));
pieChart.draw(data2, {title: 'Another Fine Pie Chart', width: 400, height: 240, is3D: true});
}
```

The example shows you how to use the Data definition URL in code.



Category ID	Freight Total
1	500
2	521
3	600
4	750
5	900
6	902
7	1020
8	1200
9	500
10	488



And the resulting output looks like the image shown above.

Though we've shown you these examples within Logi applications for convenience, you can use the same techniques with non-Logi applications or with HTML page data consumers. Just issue the URL for the Data definition to get the JSON data stream. And, of course, there are other techniques that can be used to retrieve and use the data.

Google Connections

Logi Info's **Connection.Google Docs** and **Connection.Google Maps** elements require specialized connection configurations.

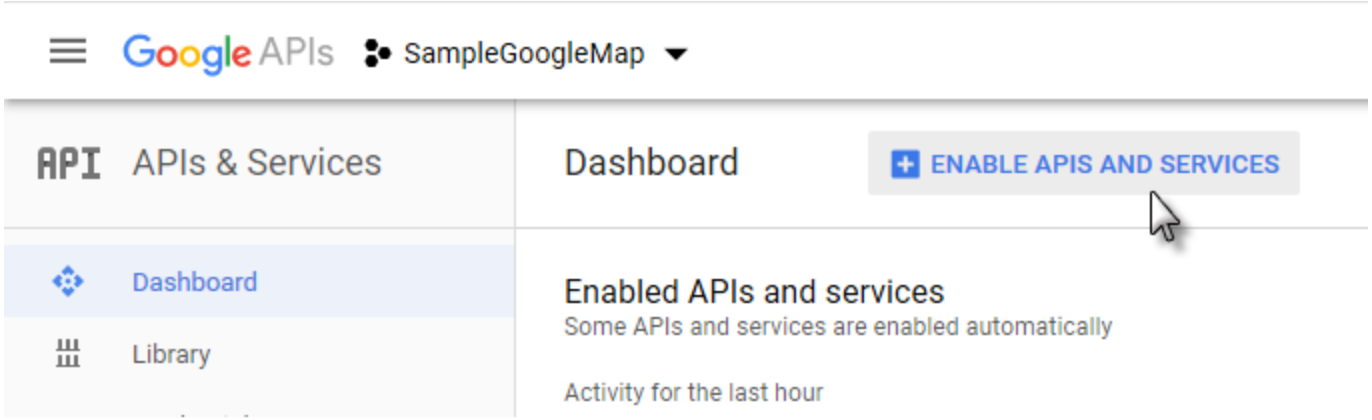
Using Geocode Columns elements? Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for the geographic data used in geocoding. This provides a free alternative to engaging with Google for the data. More information is available in "Datasource Connections" on page 10.

The following topics discuss Google Connection configurations:

Using the Google API Console

Google resources use the OAuth 2.0 security scheme and employ resource metering. To use Google services in your Logi Info application, you'll need to have a Google account, enable the correct APIs, and get security credentials. This is all done through the **Google API Console**, which is available at:

https://console.developers.google.com



Login to Google and access the console and then, in the *Dashboard* page, begin the process of enabling the necessary APIs by clicking **ENABLE APIS AND SERVICES**, as shown above.

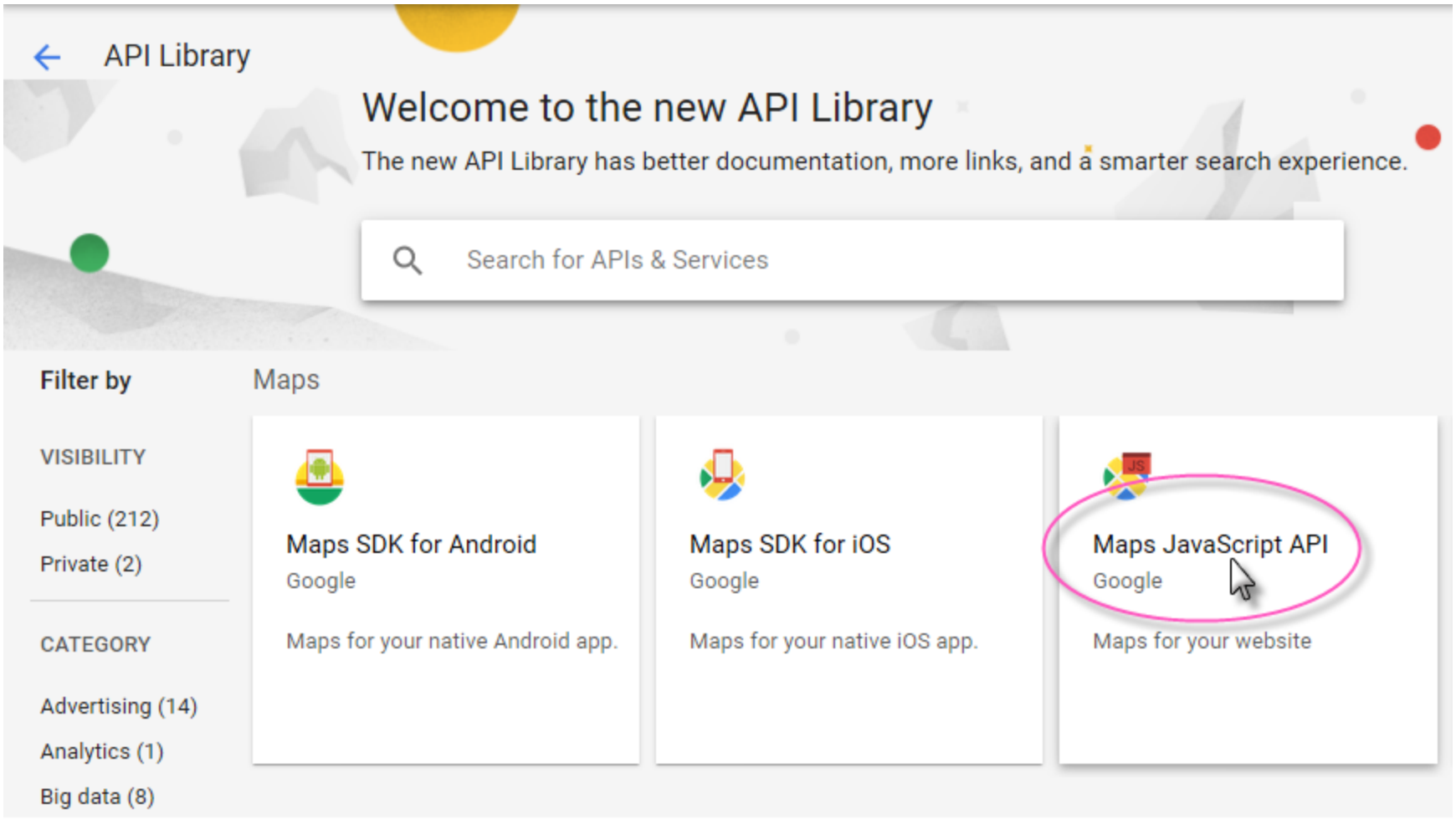
API selection and configuration details for specific resources continue in the following sections of this topic.

Configuring for Google Maps

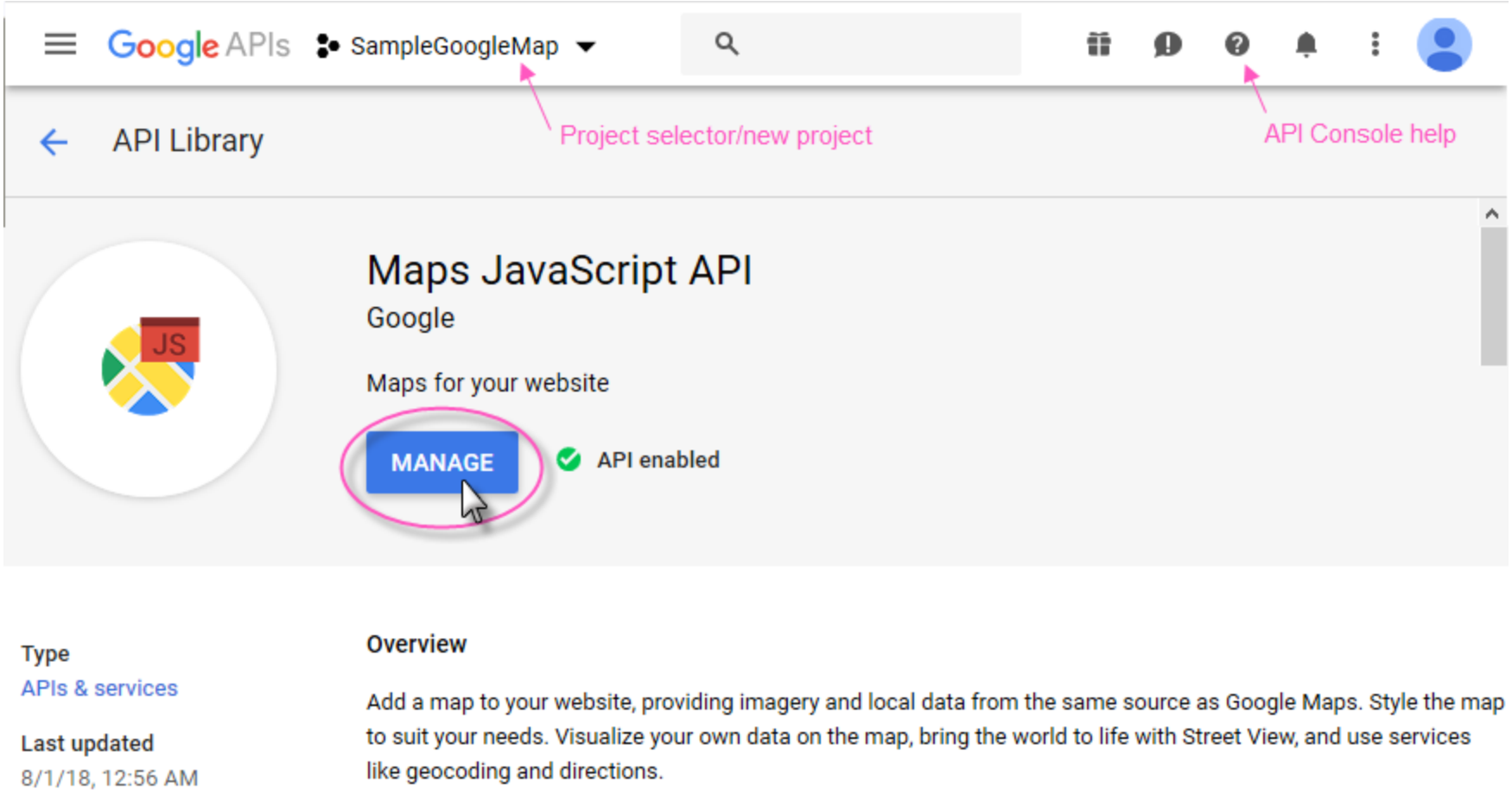
This topic describes how to configure an API Key for a Logi Info application that uses **Connection.Google Maps** and the other **Google Map** elements.

 In July 2018, Google changed its licensing scheme, ending the free API access previously offered at some usage levels. This specifically affected *geocoding* in Logi applications. If you're using Google Maps and geocoding in your Logi application you will need to:

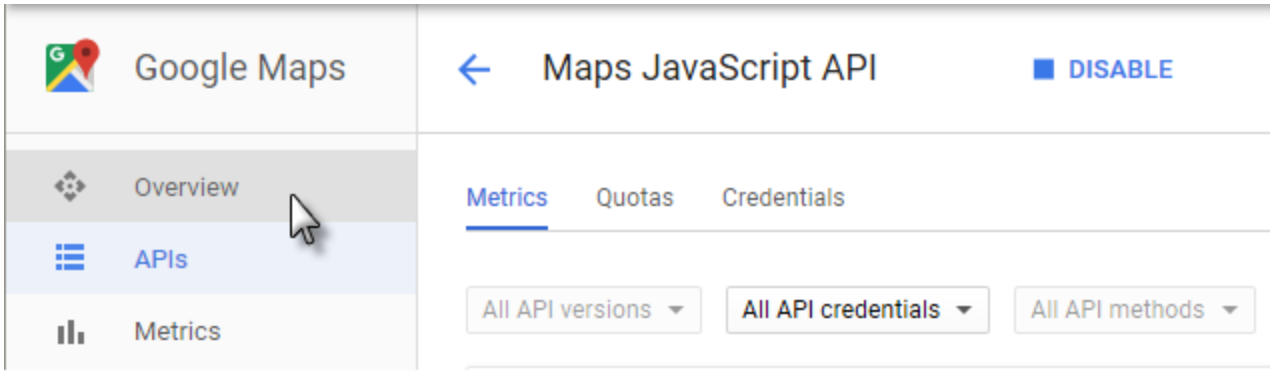
1. Review the new billing requirements outlined in Google's Pricing and Billing Changes page.
2. Enable a "billing account" for use with your Google API key.
3. Generate a new API key for use in your Logi application.
4. Upgrade your application to Logi Info v12.5 SP9 or later, and use the new key in it.



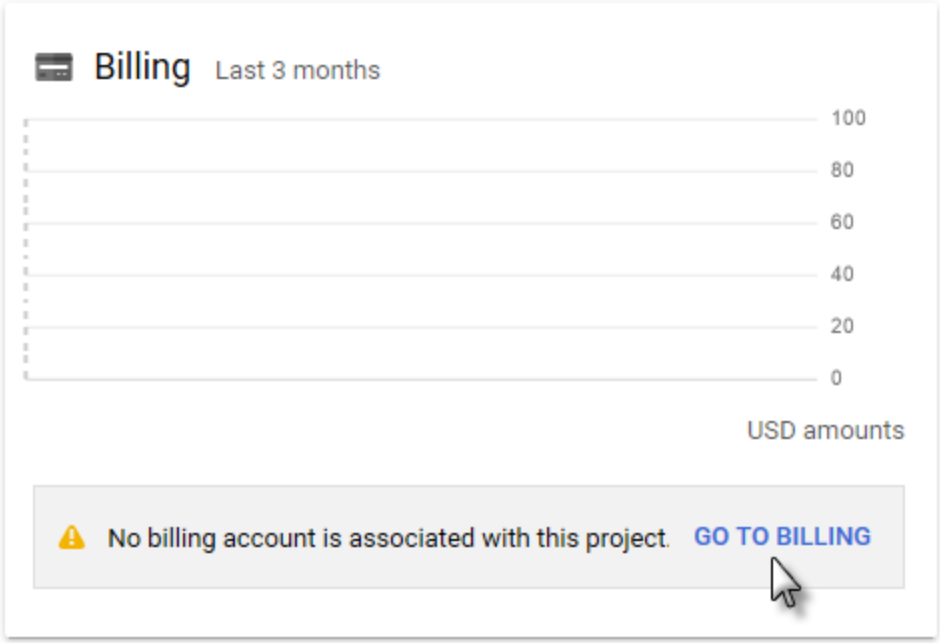
1. In the *Dashboard* page, click **ENABLE APIS AND SERVICES**, which takes you to the *API Library* page. In the Maps section, find and click the *Google Maps JavaScript API* item, as shown above.



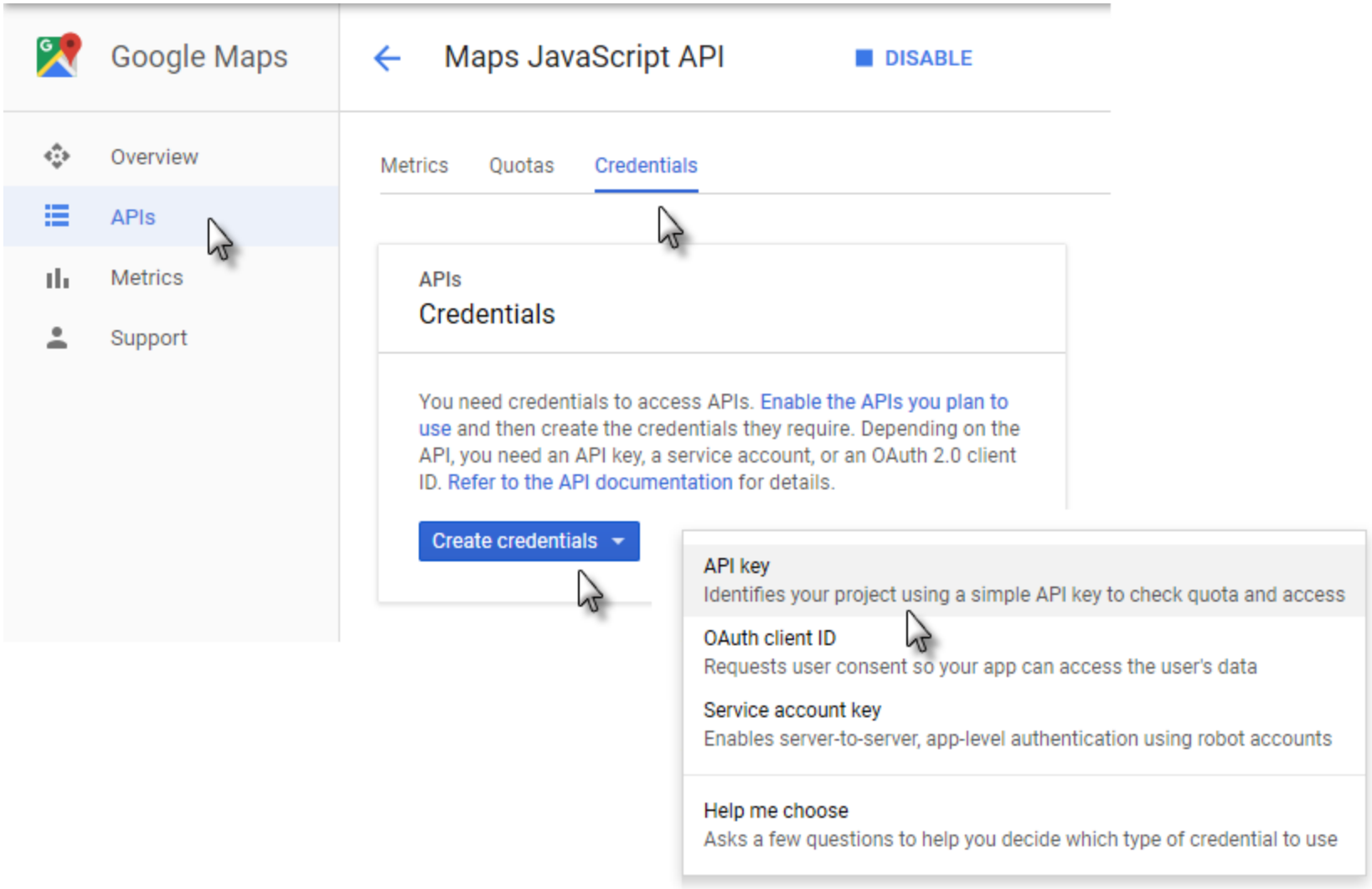
2. You'll be shown a page with information about the API, as shown above. Use the control at the top of the page to add a new project, or to select an existing project. Click **MANAGE** to configure the project.



3. The Maps JavaScript API details page will be displayed. Click the **Overview** menu item, as shown above.

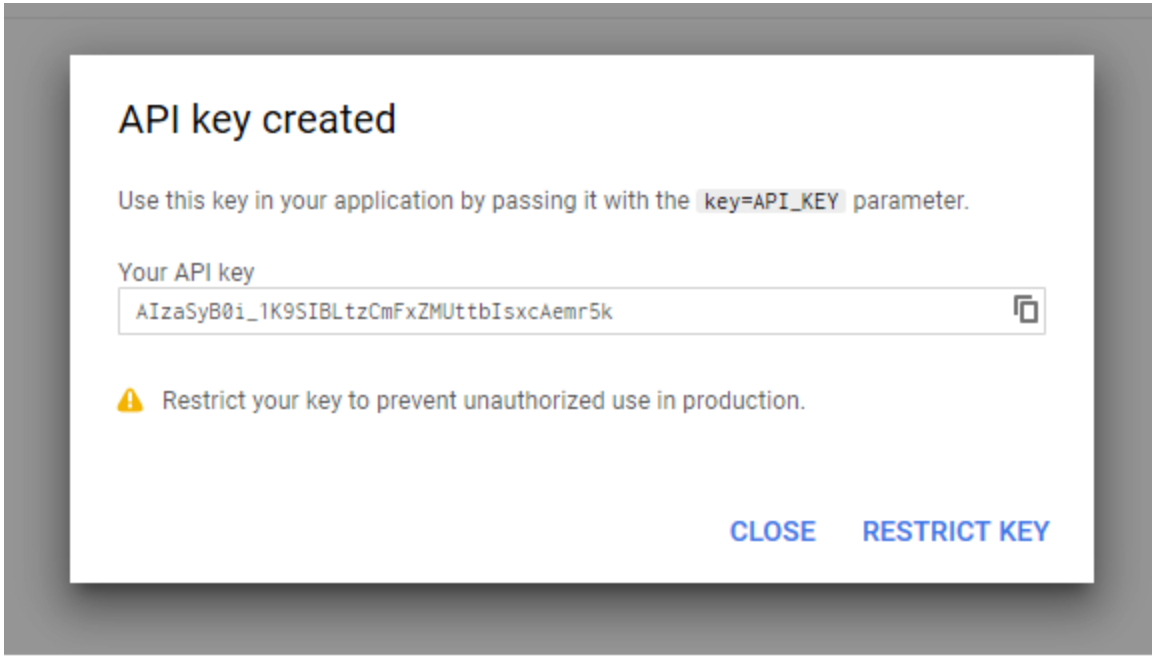


4. On the Overview page, find the Billing panel, shown above. Click the **GO TO BILLING** link to configure billing for this project. Link an existing billing account to this project, or create a new billing account and link it.

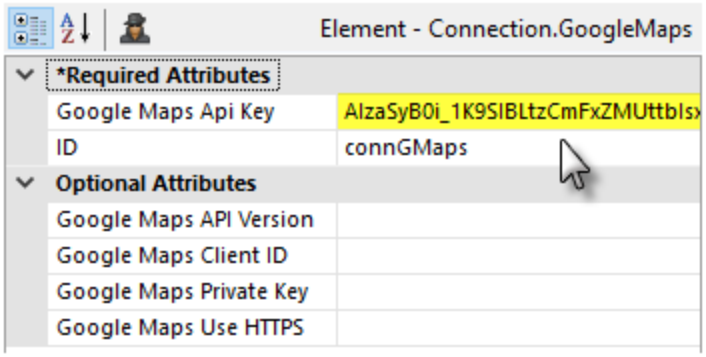


5. In the Maps JavaScript API page, select the **APIs** menu item, then the **Credentials** tab, as shown above. If you have no existing keys, click **Create credentials**, then **API Key**, as shown above. (If you have existing keys, select the one used

with your Logi application and click **REGENERATE KEY** on its details page).



6. A pop-up panel will display your new API Key, as shown above. Click the icon to copy the key to your clipboard. If desired, click **RESTRICT KEY** to provide your own key name and/or apply referrers, IP address, and other usage restrictions. 💡 It may take up to 5 minutes for this setting to take effect.



7. In Logi Studio, paste your API key into the Connection.Google Maps element's **Google Maps Api Key** attribute, as shown above. Be sure there are no leading or trailing spaces around the key. Also set the required **ID** attribute as desired.

Your connection is now configured and you can proceed to develop and test your Logi Info application with Google Maps.

The API Console's *Dashboard* page allows you to monitor the number of requests and other performance details of the Google API.

Other Connection Attributes

The Connection.Google Maps element also has these attributes:

Attribute	Description
Google Maps API Version	Specifies the Google Maps JavaScript API version to use. The API is regularly updated with new features, bug fixes, and performance improvements. If left blank, the latest version will be used. Information is available about API versions and release notes.

Attribute	Description
Google Maps Client ID	Specifies the Client ID value that Google provides to its "Google Maps API for Business" license customers. This ID is used with the **Geocode Columns** element, which works under DataLayers to geocode addresses. You will need a Google license in order to use this.
Google Maps Private Key	Specifies the Private Key value that Google provides to its "Google Maps API for Business" license customers. This key is used with the Geocode Columns element, which works under DataLayers to geocode addresses. You will need a Google license in order to use this.
Google Maps Use HTTPS	Specifies whether the Google Maps API should be accessed using HTTPS. The default value is: *False*.

Configuring for Google Docs

This topic describes the configuration for a Logi Info application that uses **Connection.Google Docs** and elements like **DataLayer.Google Spreadsheet**.

Google Docs uses OAuth 2.0 authentication and you'll need to have a Google account in order to create appropriate credentials for your Logi application. Here are the general steps that you'll need to take to configure the **Connection.Google Docs** element:

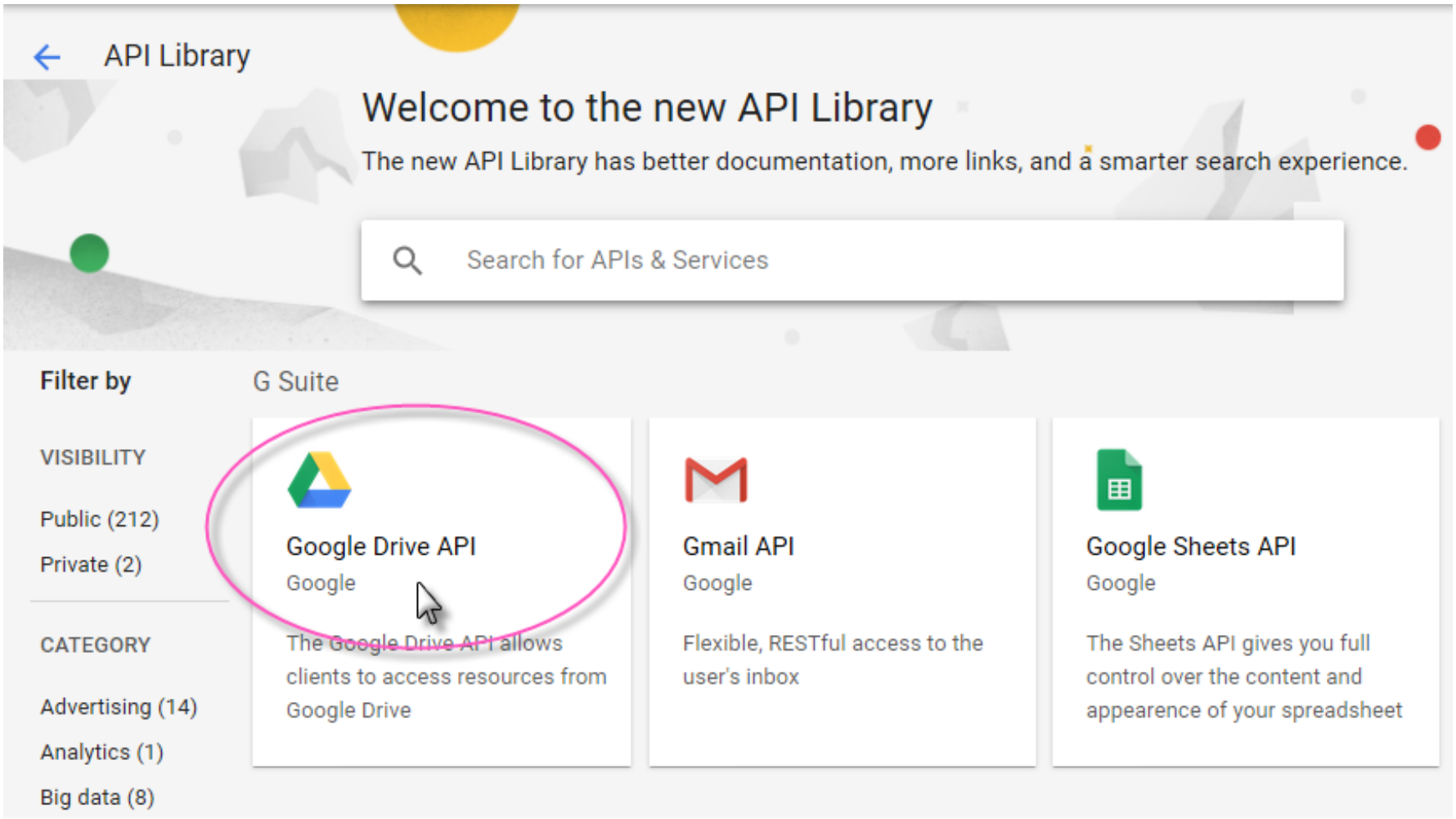
1. Login to your Google account and obtain a Client ID and Client Secret from the Google API Console.
2. Fill in the attributes (see below) of the Connection.Google Docs element, *except***GoogleOAuth2Json**.
3. Click the Browse button at the end of the GoogleOAuth2Json attribute.
4. In the Google Authorization page that opens, ensure the user name matches the Username attribute value you entered.
5. Click **Allow** on the authorization page and it will close
6. Click **OK** on the dialog box in Studio and the GoogleOAuth2Json attribute value will be filled-in for you.

Here are the Connection.Google Docs element attributes:

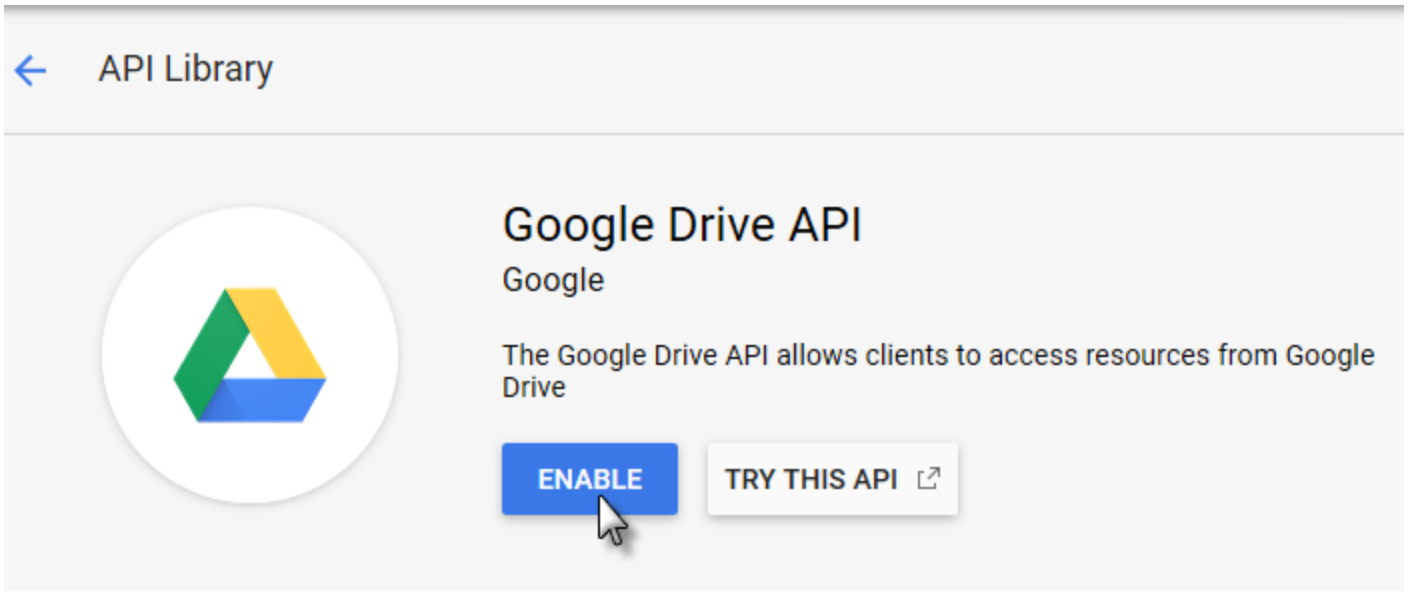
Attribute	Description
Client ID	(Required) Specifies the OAuth 2.0 Client ID value from the Google API Console.
Client Secret	(Required) Specifies the OAuth 2.0 Client Secret value from the Google API Console.
GoogleOAuth2Json	(Required) Specifies a special JSON string, generated by the Google Authorization page. *Do not* click the browse button in this attribute until all other attributes have been filled-in. See Steps 4, 5, and 6 above.

Attribute	Description
ID	(Required) Specifies a unique, arbitrary connection element ID.
Username	(Required) Specifies the user name for the Google account that owns the data we want to access.

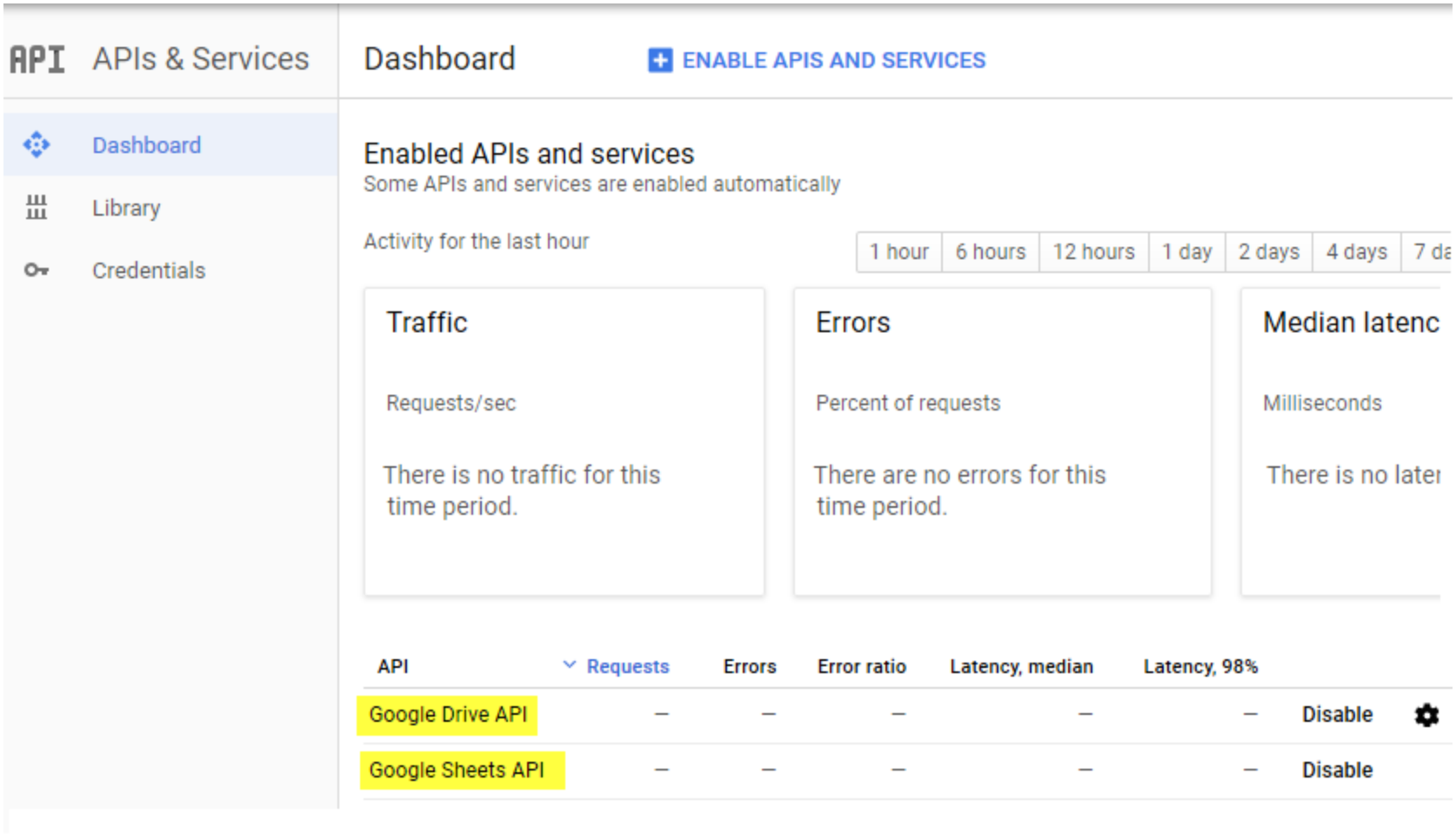
Here are the detailed instructions for your interaction with the Google API Console:



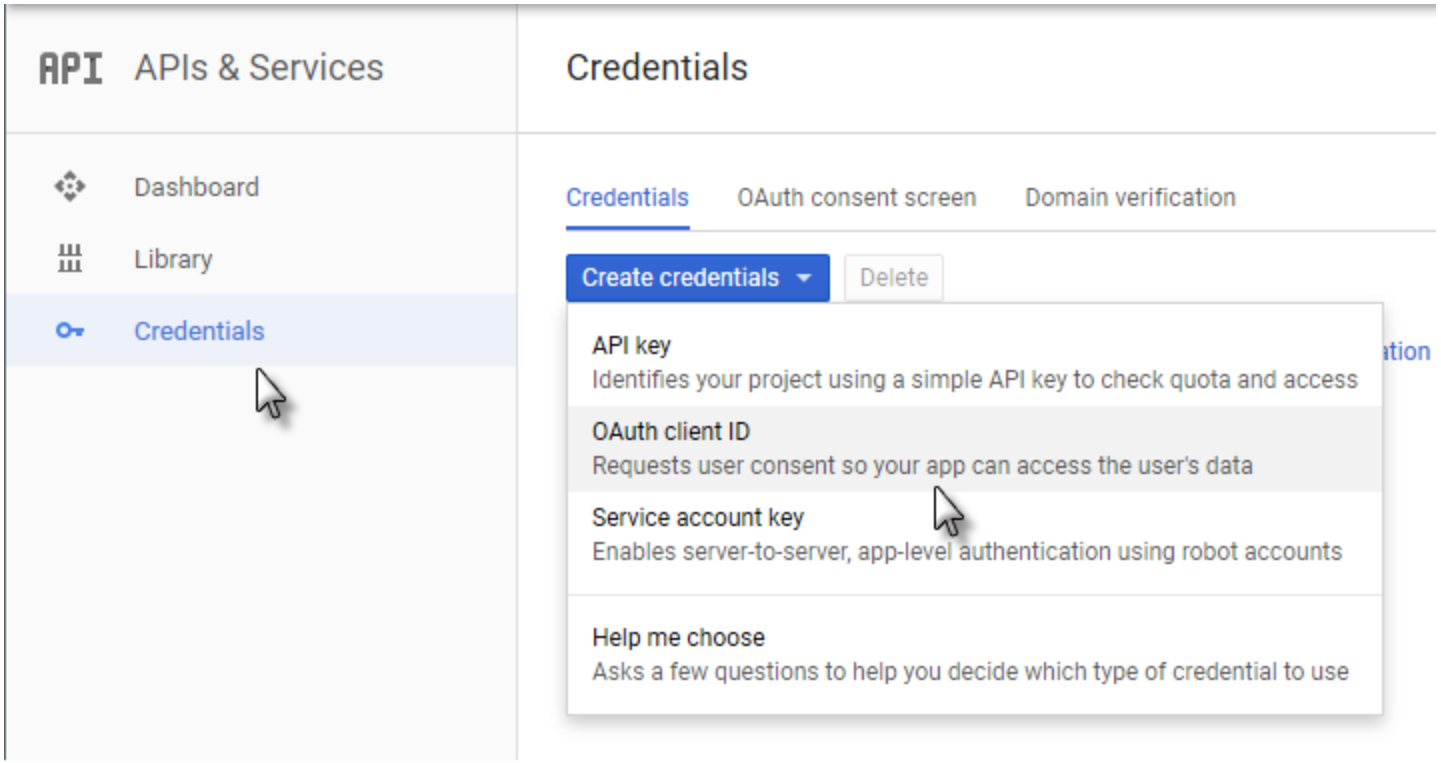
1. Click **ENABLE APIS AND SERVICES** in the *Dashboard* page, which takes you to the *Library* page. In the G Suite section, find and click the *Google Drive API*, as shown above.



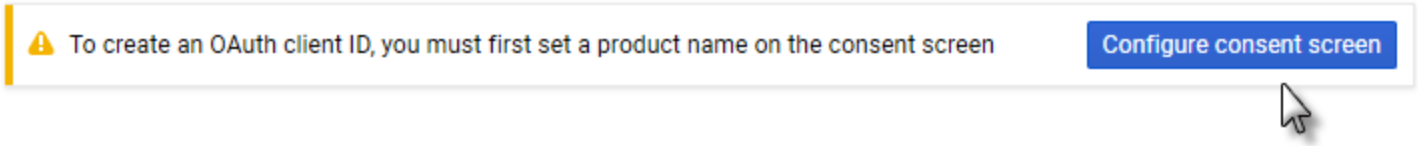
2. You'll be shown a page with information about the API. Click **ENABLE** as shown above.
3. Return to the library and repeat the process for the *Google Sheets API*.



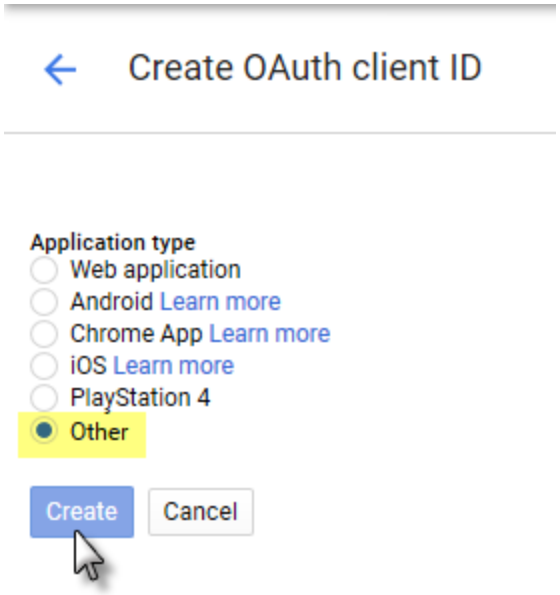
4. Return to the *Dashboard* page of the console, as shown above, and you should see the two newly-enabled APIs.



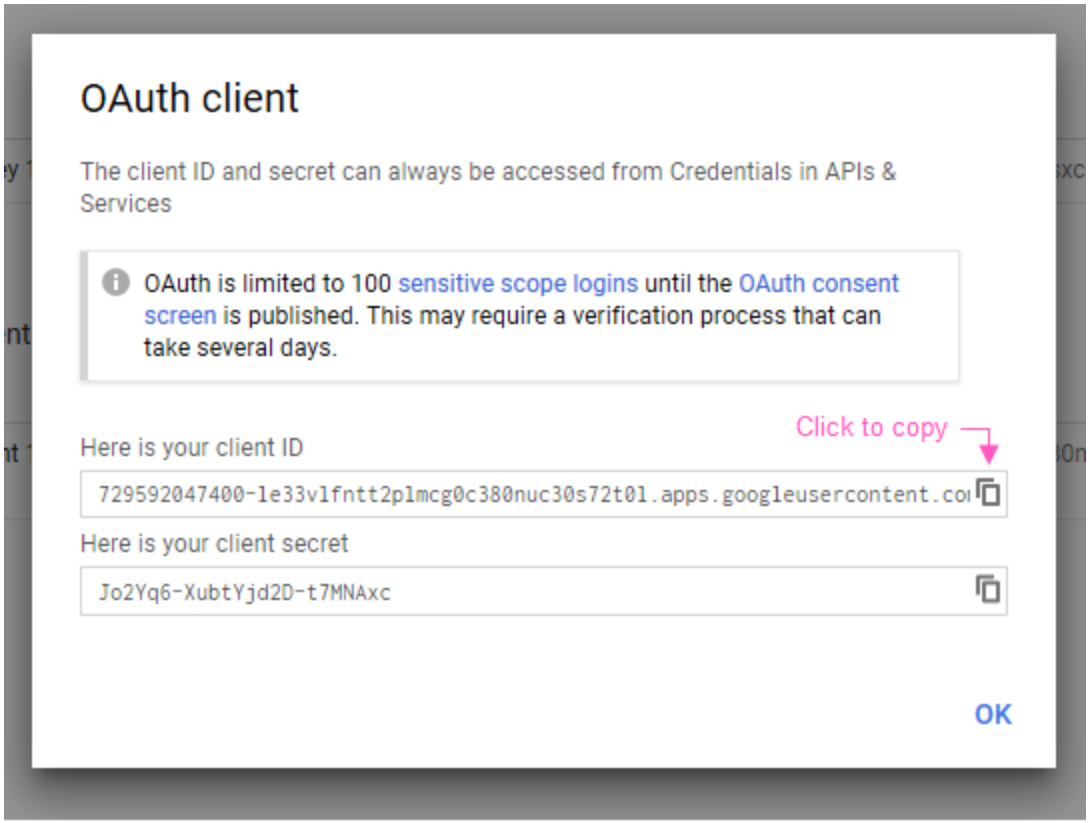
5. If you already have credentials, skip to Step 10. To create new credentials, go to the Credentials page, as shown above, and click **Create credentials**. Select the *OAuth client ID* option.



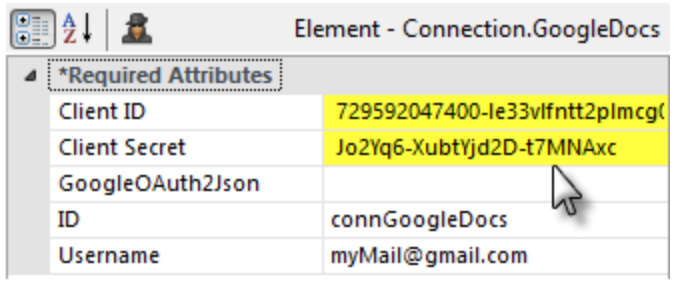
6. If you see the warning shown above, click **Configure consent screen** and provide an application name. You do not need to provide any other information. Click **Save** at the bottom of the page.



7. In the Application Type list, select the *Other* application type and provide an arbitrary name for the client. Click **Create** to continue.



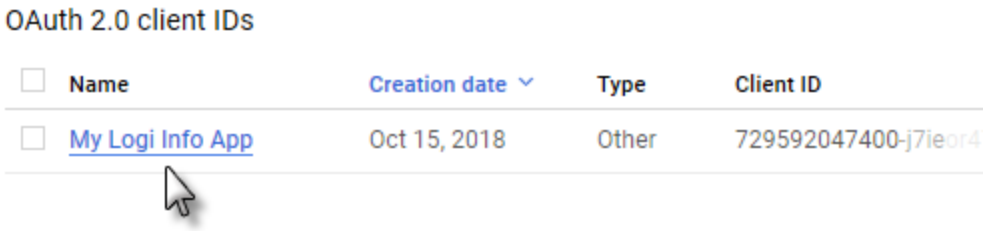
8. Your new credentials will be displayed, as shown above. Copy the values...



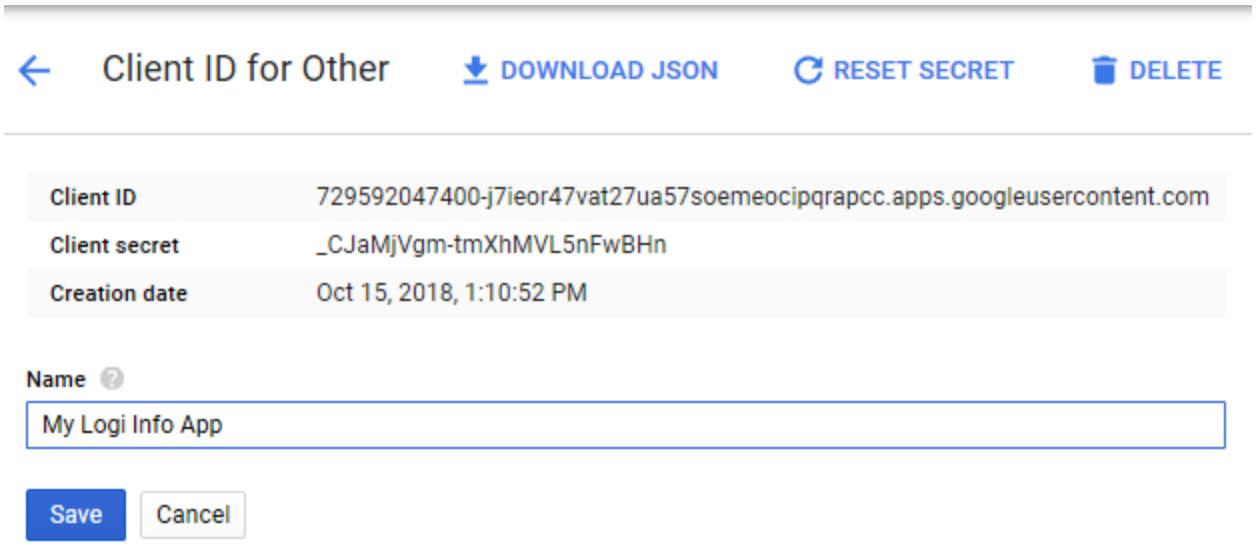
9. ...and paste them into the Connection.Google Docs element's **Client ID** and **Client Secret** attributes, as shown above.

   Be sure that there are *no leading* or *trailing spaces* in the values you paste into these attributes, especially if you use the Google-supplied Copy icons.

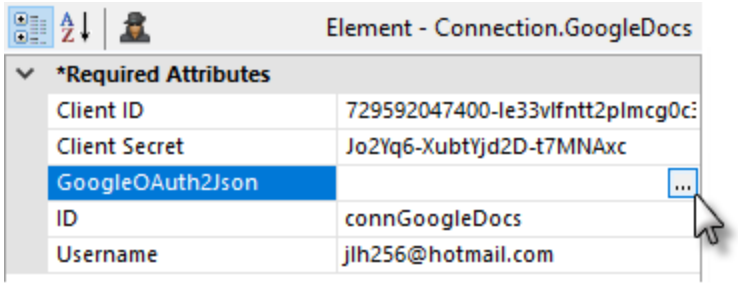
   Then provide the required element **ID** and enter the **Username** for the account that owns the data that you want to access on Google Docs. In the Google API Console, click **OK** to continue, and skip to Step 8 in these instructions.



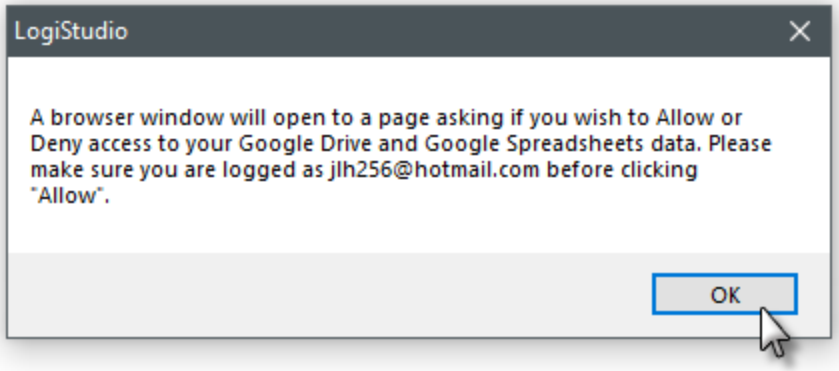
10. **If you already have OAuth client credentials** and just need the Client security values, in the Credentials page click the name of the client ID, as shown above, to see them:



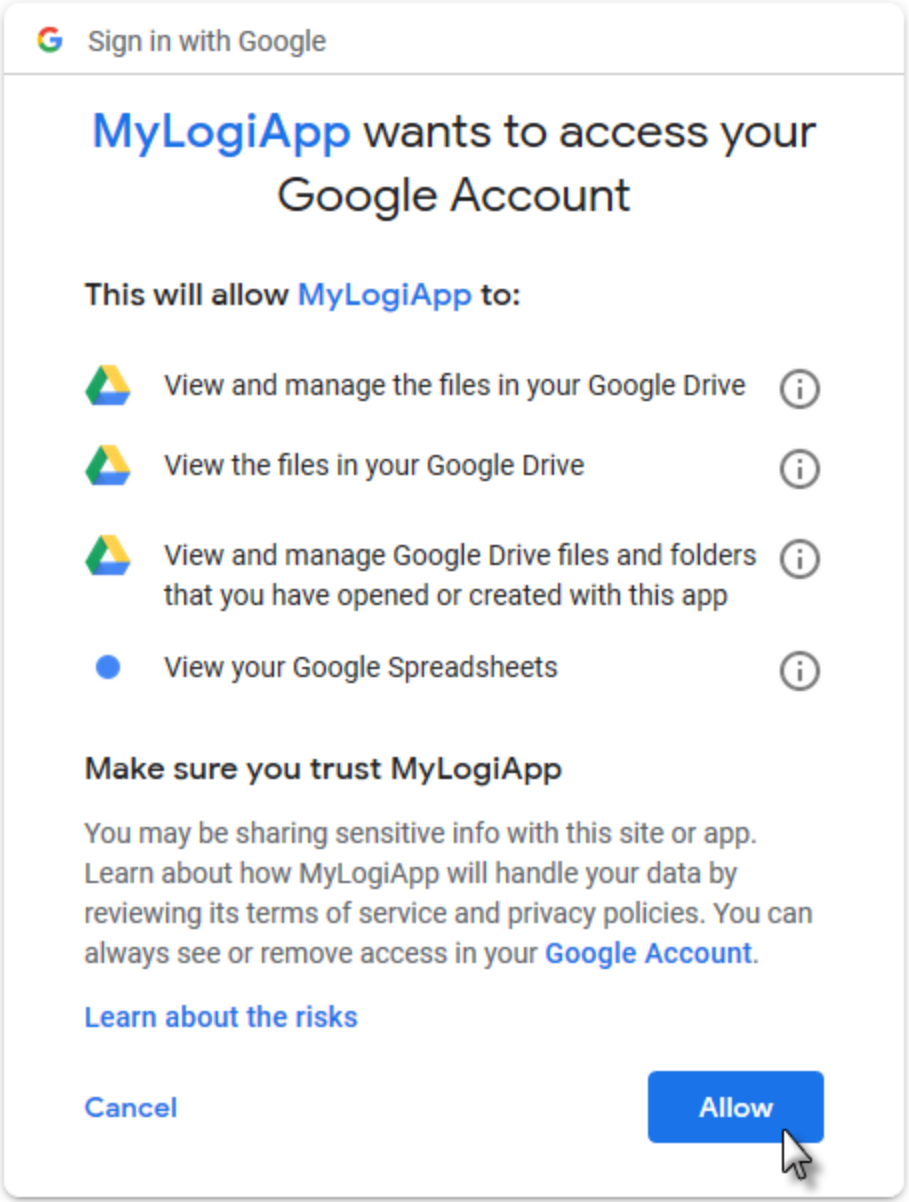
The details of the credentials set will be displayed, including the **Client ID** and **Client Secret** values. Copy and paste these values into the Connection.Google Docs element's Client ID and Client Secret attributes. You can also change the restrictions on the credentials here. Then provide the required element ID and enter the Username for the account that owns the data that you want to access on Google Docs.



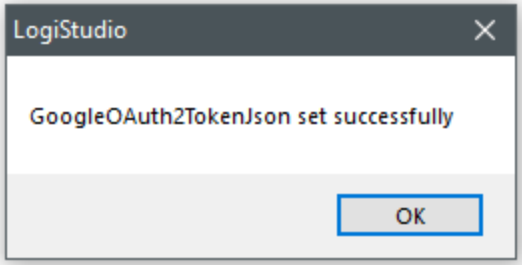
11. Next, in Logi Studio, click the Browse button in the Connection.Google Docs element's **GoogleOAuth2Json** attribute, as shown above. If you don't see the button, click the blank attribute value first. This dialog box will appear:



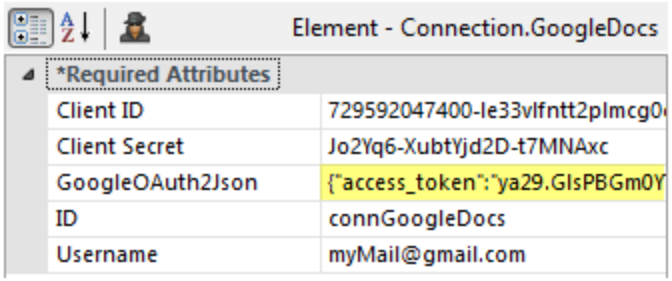
Click **OK** to continue.


Sign in with Google
MyLogiApp wants to access your Google Account
This will allow MyLogiApp to:
View and manage the files in your Google Drive
View the files in your Google Drive
View and manage Google Drive files and folders that you have opened or created with this app
View your Google Spreadsheets
Make sure you trust MyLogiApp
You may be sharing sensitive info with this site or app. Learn about how MyLogiApp will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.
Learn about the risks
Cancel
Allow

12. Your browser will open, requesting a login to Google. After a successful login, the page shown above will be displayed; click **Allow** to consent to access.



13. In Logi Studio, the dialog box shown above will appear. Click **OK** to continue.



Now when you look at the Connection element, the GoogleOAuth2Json attribute should have a JSON value.

You're now ready to use your connection to retrieve data from Google Docs.

Oracle Packages

Oracle does not support directly returning **row sets** from stored procedures; however, this functionality is possible using Oracle Packages.

In the past most Oracle Server databases contained only a small amount of code in stored procedures but, today, this is rapidly changing. **Oracle Packages** serve as a way to group **stored procedures** and **functions** together, typically based upon their common role. Benefits include better performance, code isolation, and the close coupling of data with behavior.

Logi reporting products support calls to Oracle Package procedures using **DataLayer.SQL**, **Procedure.SQL**, **DataLayer.SP**, and **Procedure.SP** elements.

When working with Oracle, you will need to ensure that you have the correct Oracle Provider (if using OLEDB) or the Oracle Client (if using ODBC) installed on the web server.

The following topics discuss the use of Oracle Packages:

- Writing Oracle Packages and Procedures
- The Oracle Connection String
- Using DataLayer.SQL & Procedure.SQL
- Using DataLayer.SP & Procedure.SP

Logi Info and SSRM now support Oracle 19c databases.

Writing Oracle Packages and Procedures

Oracle stored procedures can perform **non-query activities** such as INSERTs, UPDATEs, and DELETEs that do not necessarily return data.

However, in order for a stored procedure that *does* return data to work with Logi applications, it must return a valid **result set**, such as the output from a SELECT statement; data **cannot** be returned as individual values in **output parameters**. The package and procedure should be similar to this example, which returns a list of UserIDs from a table in the database :

```
PACKAGE LGX_GET_DATA
AS
TYPE UserCur IS REF CURSOR;
PROCEDURE AllUser(o_EmpCursor OUT UserCur);
END LGX_GET_DATA;

PACKAGE BODY "LGX_GET_DATA"
AS
PROCEDURE AllUser(o_EmpCursor OUT UserCur)
AS
BEGIN
OPEN o_EmpCursor FOR
SELECT USER_ID
FROM USER_ACCOUNTS
ORDER BY USER_ID;
END AllUser;
END LGX_GET_DATA;
```

Your stored procedure can have any number of input parameters but it can have only **one output cursor**. In addition, the output cursor must be **last** in your procedure's **argument list**. For example, this **will work**

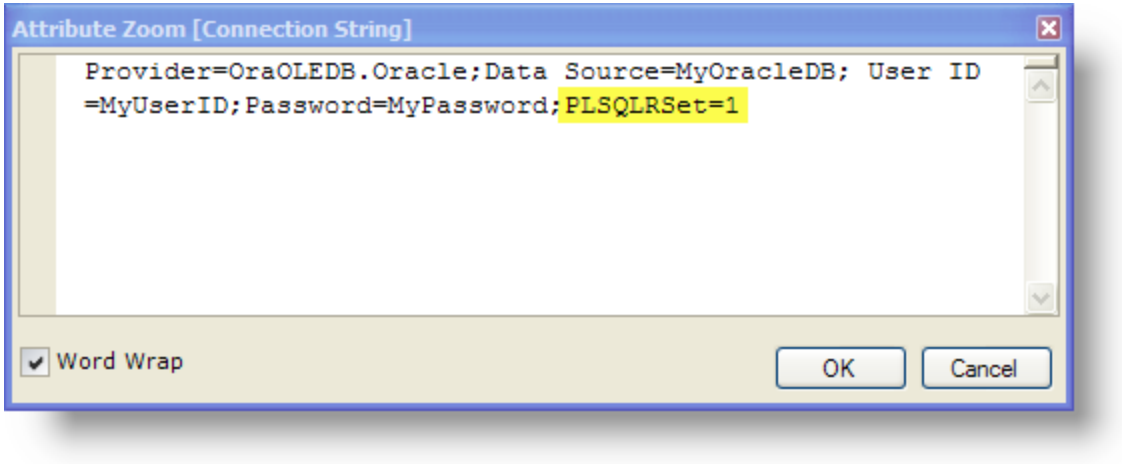
```
procedure X( iParam1 in integer, iParam2 in integer, ..., o_cur out ioCursor )
```

but this **will not**:

```
procedure X( o_cur out ioCursor, iParam1 in integer, iParam2 in integer, .... )
```

The Oracle Connection String

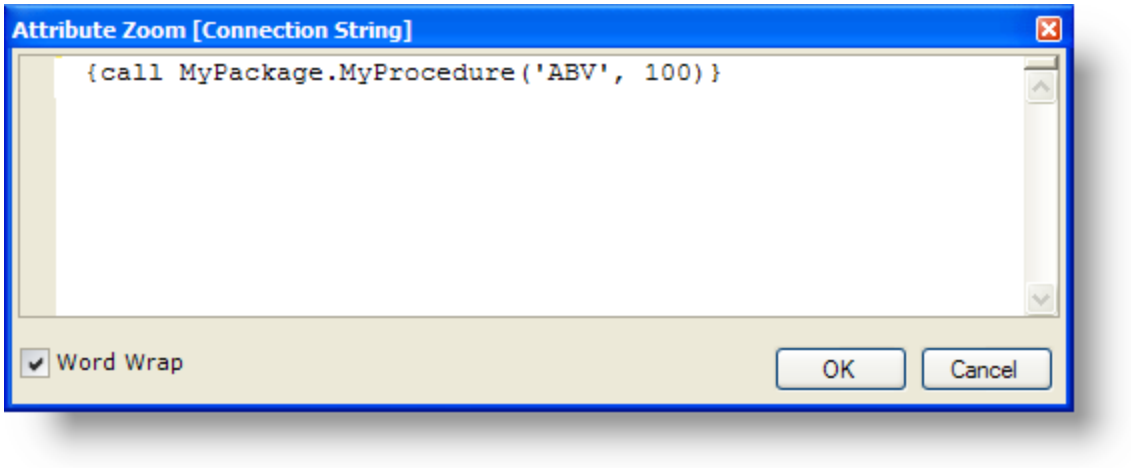
The Connection String for an Oracle database requires a **special parameter** that you must code manually.



Add the following parameter, highlighted above, to your Connection element's **Connection String** attribute: `PLSQLRSet=1` This informs the OLEDB provider that row sets can be returned (in the example above, an OLEDB provider supplied by Oracle has been installed and is used). Check here to see examples of Oracle connection strings for a variety of configurations and circumstances. All must include the parameter shown above in order to return row sets.

Using DataLayer.SQL or Procedure.SQL Elements

Although they're generally not used with stored procedures, the **DataLayer.SQL** and **Procedure.SQL** elements provide an easy way to call a stored procedure in an Oracle Package.



As shown above, if you use the DataLayer.SQL or Procedure.SQL elements, their **Source** attribute should contain a command with the following syntax:

```
{call PackageName.ProcedureName('StringValue',NumericValue,...)}
```

The data in the returned row set is referenced using @Data tokens as usual. **Usage Example**: imagine that a stored procedure named *MyProc* has been included an Oracle Package named *MyPackage*. The stored procedure requires two parameters: one a string and the other a number.

To use the stored procedure, the DataLayer element's Source attribute should contain:

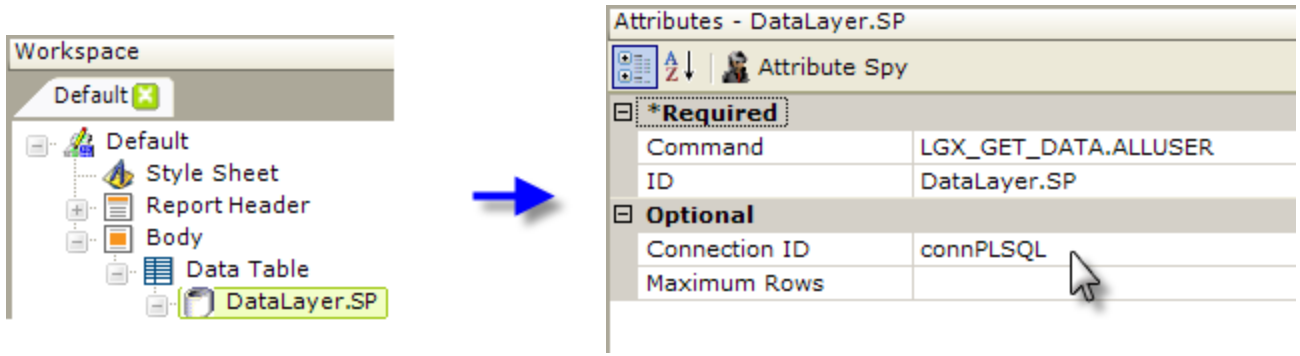
```
{call MyPackage.MyProc('George',100)}
```

If you want to call this stored procedure with variable arguments using token, the Source attribute should contain:

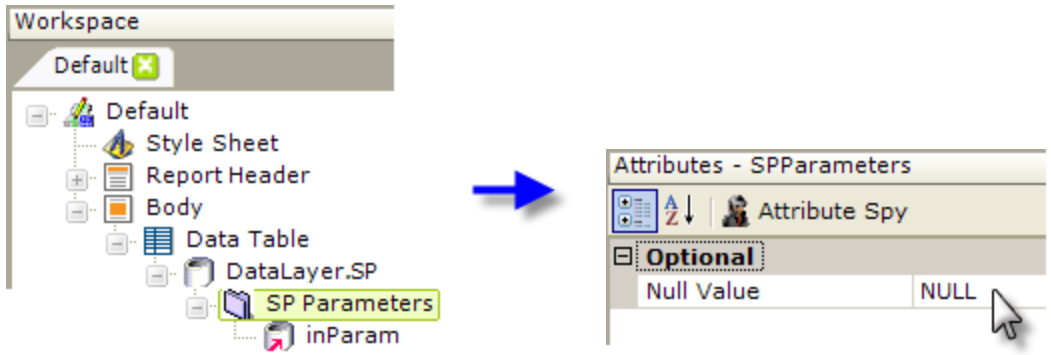
```
{call MyPackage.MyProc('@Request.FirstName~',@Request.Count~)}
```

Using DataLayer.SP or Procedure.SP Elements

The Logi Studio elements designed to be used with stored procedures, **DataLayer.SP** and **Procedure.SP** can be used to retrieve data from an Oracle Package procedure. However, they behave slightly differently than they do in other circumstances.



The DataLayer.SP or Procedure.SP element's **Command** attribute uses a particular syntax, shown above, to identify the Oracle Package and Procedure. This is *<packageName>.<procedureName>.*



As shown above, set the SP Parameters element's **Null Value** attribute to NULL. You may use as many **input** parameters in your definition as you wish but **do not** use any **output parameters**. The data in the returned row set is referenced using @Data tokens as usual.

The Lookup Element

The **Lookup** element is used with datalayer elements to perform a "look up" of related data. It runs its own datalayer once for each row in its parent datalayer and joins the results. Unlike a typical join, it doesn't require that all of the target data be downloaded before the join occurs. This topic discusses use of this element.

The following sections are covered in this topic:

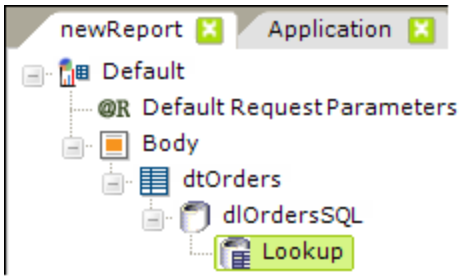
About Lookup

The **Lookup** element is available for use with all datalayer elements. It's particularly useful and provides improved performance when trying to join small data sets to very large data sets. For example, when trying to correlate the IDs and names of a few dozen sales people with the employee IDs found in thousands of sales records.

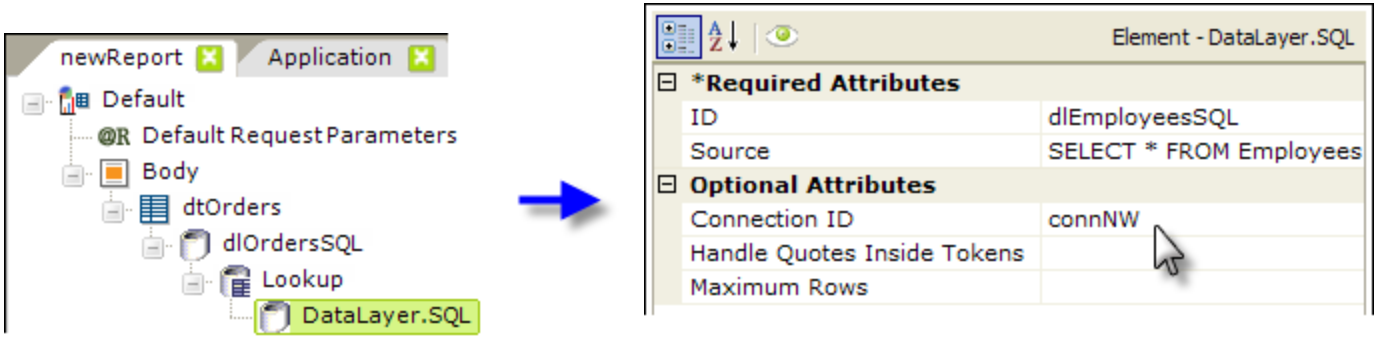
In operation, the element basically runs its child datalayer once *for each row* in its parent datalayer. In other words, it loops through its entire parent datalayer, performing for each row a "look up" in its child datalayer for matching data. Logi developers who work with Process tasks may recognize this as behavior analogous to the operation of the Procedure.Run DataLayer Rows element.

Using Lookup

The following example illustrates how the **Lookup** is used:



1. As shown above, a **Lookup** element is added as a child to a datalayer element. There are no required attributes for the element.



2. Beneath it, add a child datalayer, as shown above. In the example, it's a **DataLayer.SQL** element and its **Source** attribute will use a token (which refers to the data in the parent datalayer) in its query for the correct Employee record:

```
SELECT * FROM Employees WHERE EmployeeID = @Data.EmployeeID~
```

If the child datalayer is a different type that doesn't support a query, such as DataLayer.XML File, use the @Data token from above in a Condition or Compare Filter to get the correct data. Note, however, that you must use a special "nested" @Data

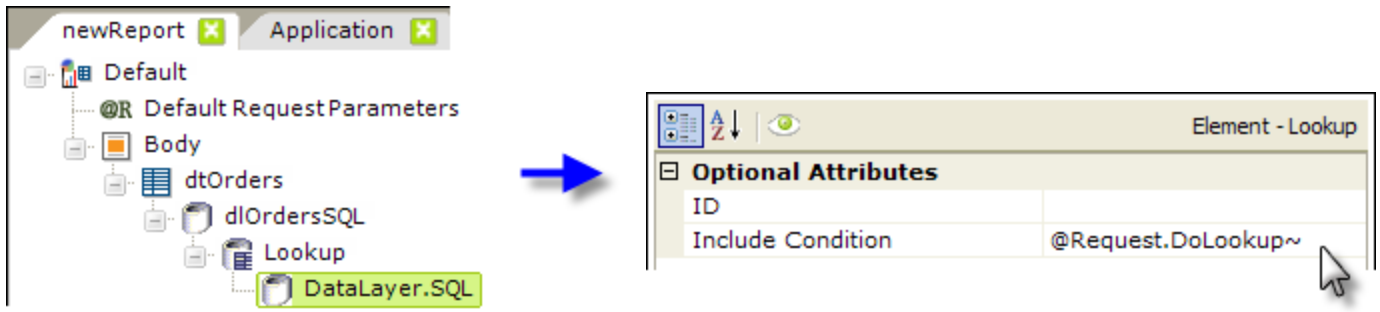
token. Basically, this is a token-within-a-token; the inner token gets evaluated first, then the outer token. It looks like this:

```
@Data.@Request.column_name~~
```

Yes, it has two @ signs and two tildes. The Request token gets evaluated first, providing the column name for the @Data token. This is necessary due to the order in which the Logi Engine processes elements.

After the look up is complete, the data found by the Lookup element will be available for display, etc. using regular @Data tokens.

The Lookup element also has an **Include Condition** attribute:



If the value of this attribute is left blank or Lookups a formula that evaluates to *True*, the element is applied to the datalayer. If the value evaluates to *False*, the element is ignored and does not affect the datalayer. This powerful feature allows developers to dynamically determine if the Lookup element will be used or not.

Web Metadata Builder

The **Web Metadata Builder** is a tool used to create and manage "metadata files", which are used with elements that have been extended to use the Active Query Builder for data retrieval.

The following topics provide further instructions for using the tool:

- Using the Web Metadata Builder
- Managing Data Source Connections
- Managing Metadata Definitions
- Creating Multiple Metadata Files

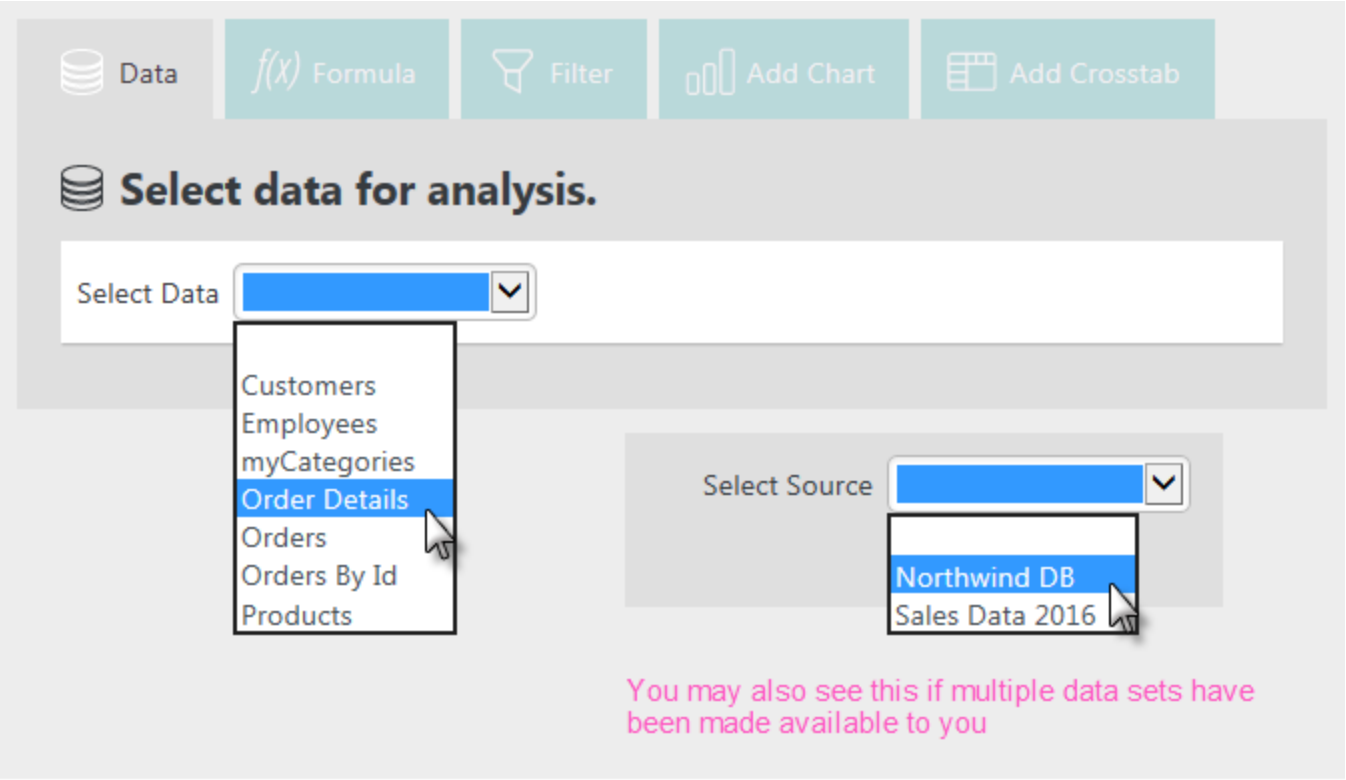
Prior to Logi Info v12.5, metadata builder functionality was available as a wizard built into Logi Studio. Beginning with v12.5, the Web Metadata Builder replaces that wizard and appears in a window within Studio.

About the Web Metadata Builder

The **Web Metadata Builder** (WMB) is a browser-based tool for building metadata. It's part of Logi Studio (v12.5+) and is also delivered with some Logi Info add-on modules.

The WMB creates and manages metadata files, which are utilized in a Logi Info application with the **Active Query Builder** and **Metadata** elements. These elements enable a UI that allows end-users to decide which data set to work with, by giving them a way to interactively select tables and joins, at runtime.

For example, an Analysis Grid (AG) is often configured with a SQL datalayer, which retrieves data based on a SQL query hard-coded by the developer. The query determines the basic dataset that will be available to all AG users for analysis.

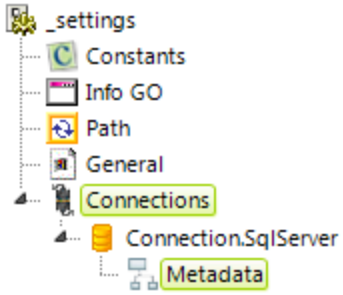


However, when added as a child of an Analysis Grid, the Active Query Builder element causes a special "Data" tab to appear at the top of the AG, as shown above. In it, users can select the tables and joins they want to work with. Intelligent assistance is provided to help users make reasonable joins and the SQL query is then created dynamically. This allows users to have a greater choice in the data they work with.

Controlling the Metadata

You, as the developer or InfoGo data manager, still exercise control over what tables, views, columns, and relationships are available for use with the Active Query Builder. This is where the "metadata file", associated with a Connection element, comes in. It enumerates all of the database objects that will be available to users for selection in the Analysis Grid. The Active Query Builder is then pointed to one or more metadata files. Multiple metadata files can be associated with a Connection element.

The WMB is used during development or InfoGo data management to create and manage metadata files. It provides the same functionality as the wizard with the same name in Logi Studio but, as a web-based tool, without the need to run Studio.



Using the WMB results in a Metadata element being inserted into the application's _Settings definition, as a child of the specified Connection element, as shown above. The Metadata element is configured to use the metadata file created using the WMB. You can, of course, manually insert and configure the element.

The *Definition Modifier Files* element is now available as a child of the Metadata element and is intended for use supporting translations and internationalization. *It should not be used in this context for other purposes, such as modifying tables or applying security.*

The Active Query Builder and Metadata elements installed with the SSRM work with these database servers:

- DB2
- Microsoft SQL Server 2005+
- MySQL
- Oracle
- Progress OpenEdge
- PostgreSQL
- Redshift (Amazon)
- Vertica (HP)

and any JDBC-accessible databases that use the same SQL syntax as one of the listed databases.

Controlling Web Metadata Builder Access

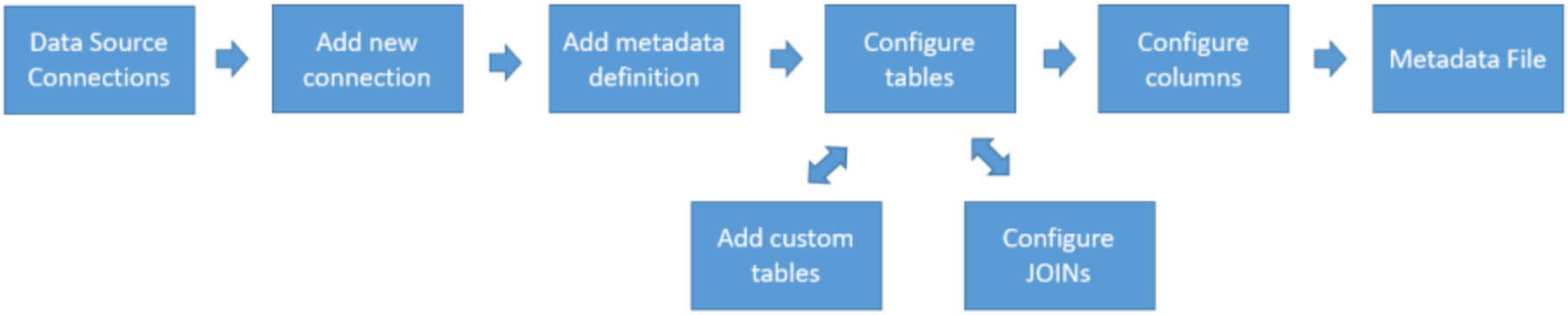
If you're *not* using Logi Security, then access to the WMB is available to anyone.

However, access to the WMB can be disabled for *all* users by setting the **General** element's **Disable Metadata Builder** attribute equal to *True*, in your _Settings definition.

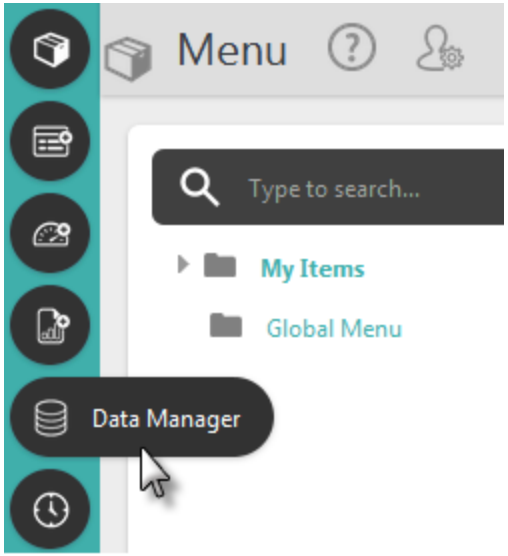
If you *are* using Logi Security, then access to the WMB can be selectively controlled by the Security element's **Metadata Admin Security Right IDs** attribute, in your _Settings definition. This attribute specifies a comma-separated list of Security Right IDs that will be granted access to the WMB. The default value is `rdMetadataAdmin`. To grant access to the WMB, either give users the `rdMetadataAdmin` security Right, or set this attribute to an existing administrator security Right.

Using the Web Metadata Builder

In addition to inserting the necessary elements, the WMB lets you identify the metadata that will be written, at the end of the process, into a *metadata file*, and to manage those files. It helps you create new metadata files or edit existing files.



The diagram above shows the process for adding a new connection and creating a metadata file. It's a simple process but provides a lot of flexibility and customization opportunities.

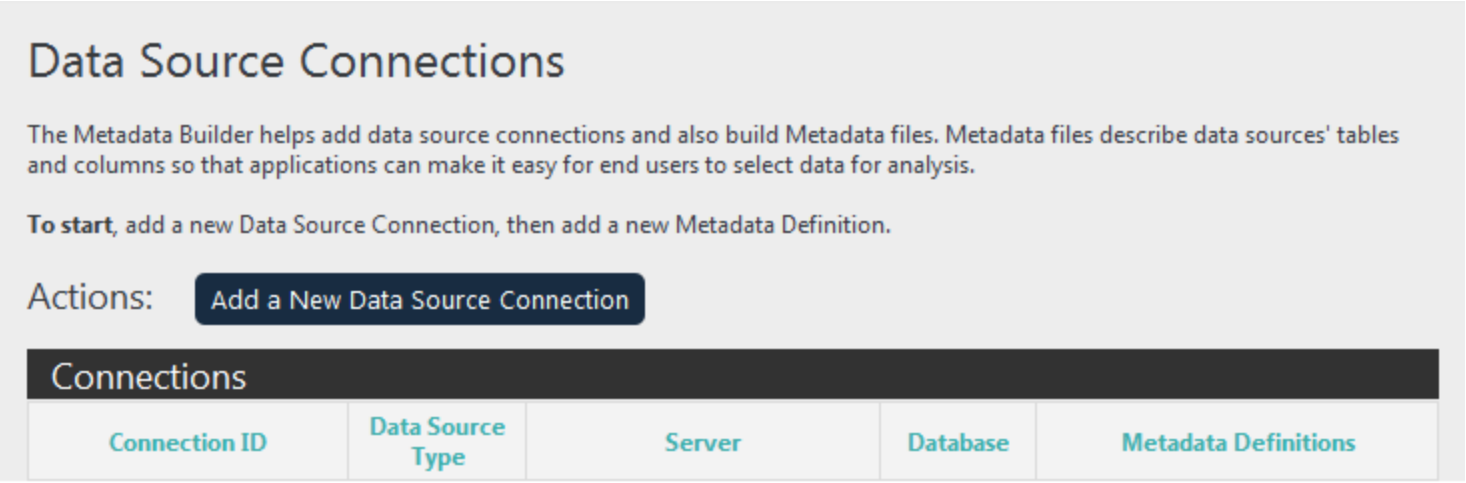


If you're using a 12.2+ version of the InfoGo application and have data management rights, in the main menu you'll see the Data Manager icon, as shown above. When you click it, the WMB will appear, embedded inside the framework and called the "Data Manager".

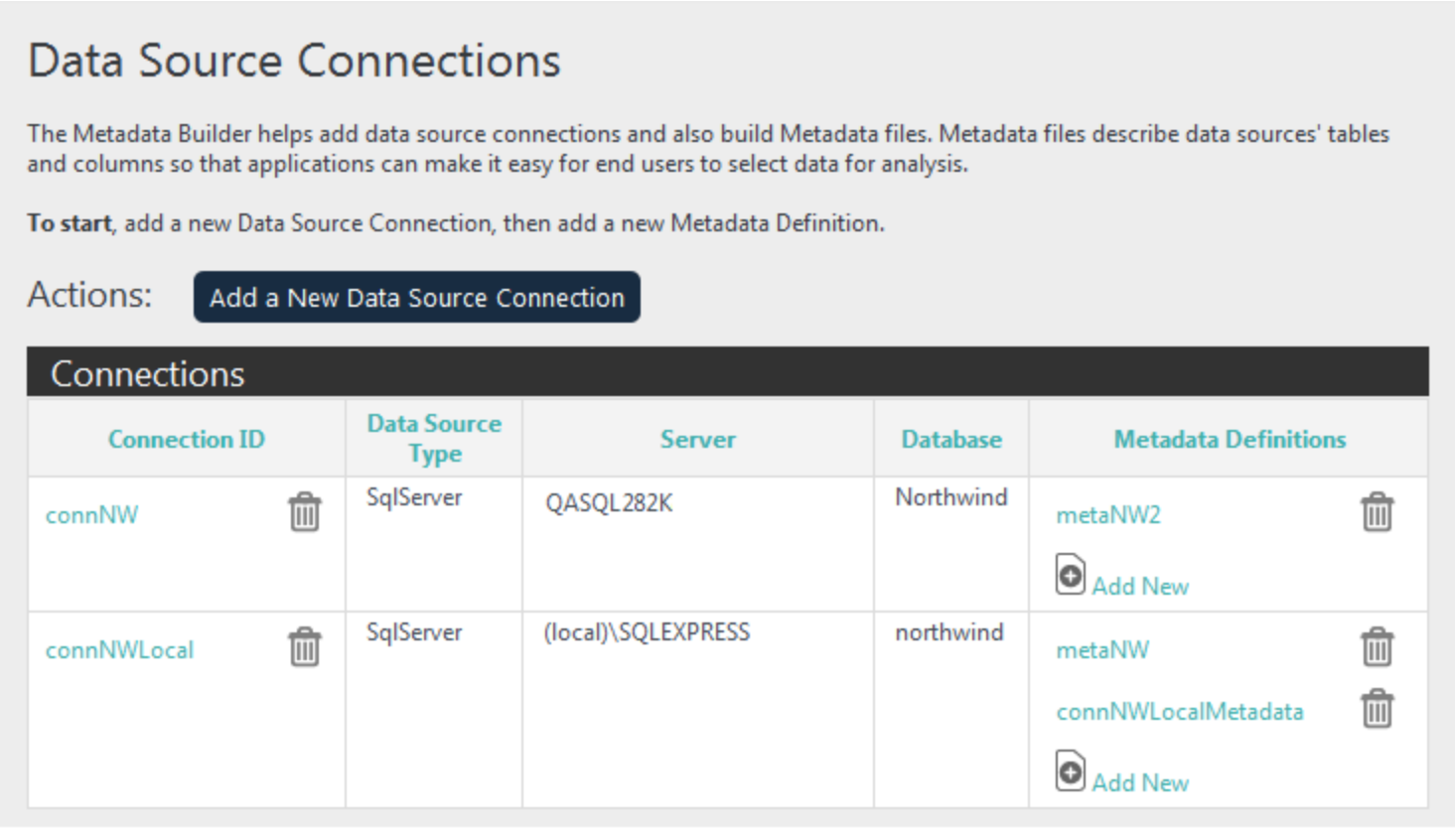
If you're *not* using InfoGo, but have installed a 12.2+ version of the SSRM and you've created a Logi application and access is enabled, you can use your browser to open the WMB directly, with this URL:

```
<yourLogiApplication>/rdPage.aspx?rdReport=rdTemplate/rdMetadata/Connections
```

In either case, you'll see this page:



If you have no Connection elements defined in your _Settings definition, the page will look like the example shown above.



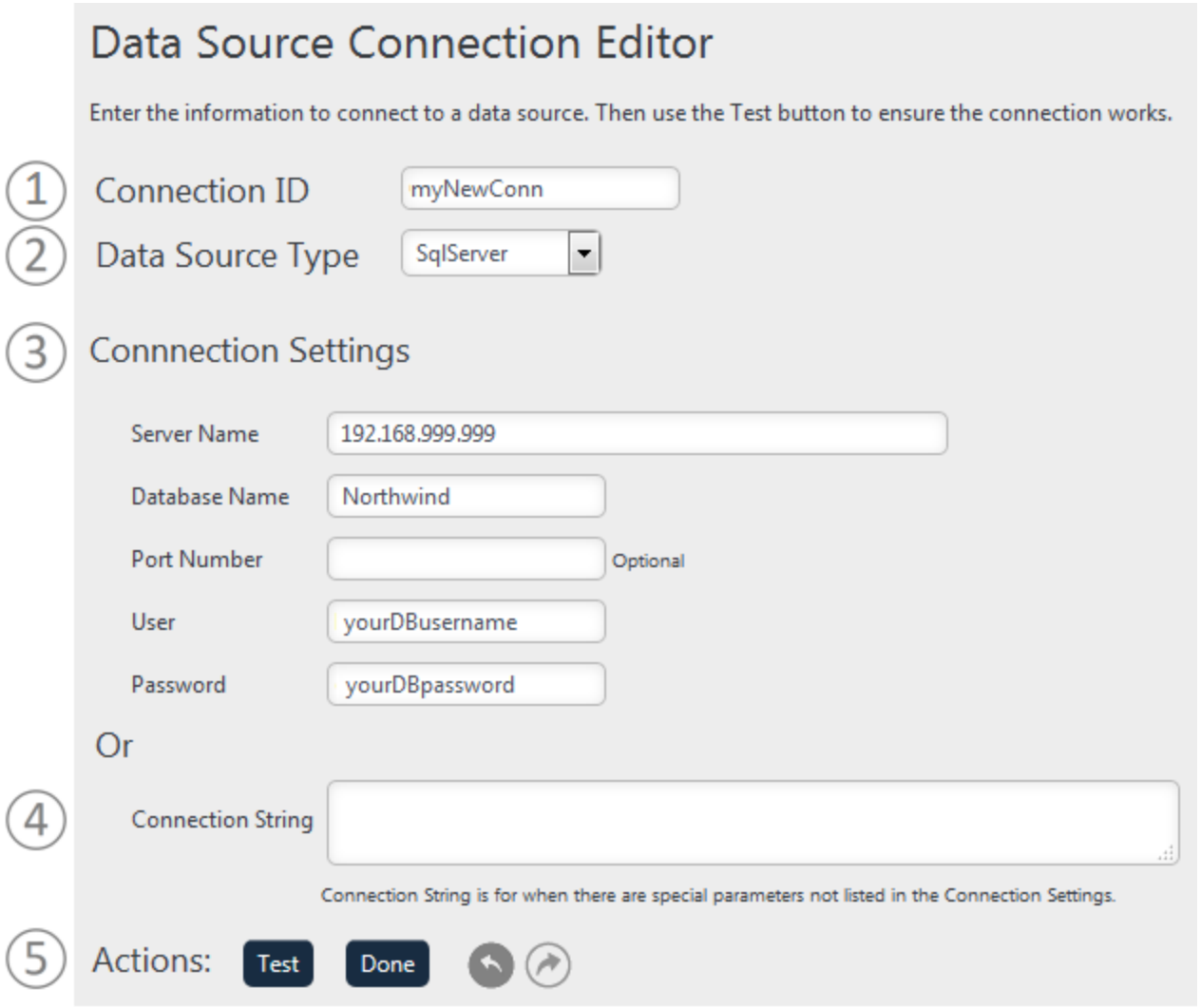
If you have existing connections, you'll see them listed on the page in a table, as shown above.

There may be small differences between the images shown here and your installation, depending on SSRM version.

Managing Data Source Connections

This topic demonstrates how to manage data source connections with a Web Metadata Builder.

Click **Add a New Data Source Connection** to add and configure a new data source connection. Click **OK** in the confirmation panel that appears.

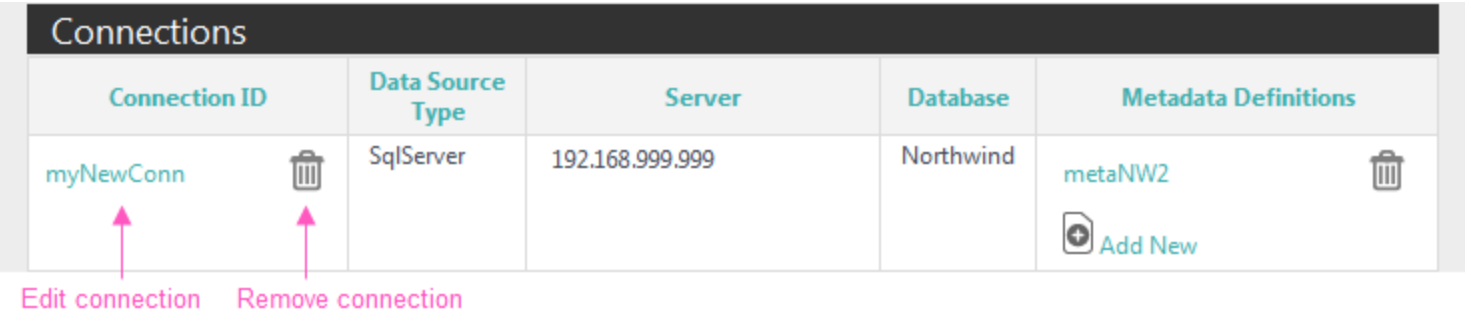


Configure a new connection as follows:

1. **Connection ID** - Give the connection a unique ID.
2. **Data Source Type** - Select the desired data source type from the drop-down list of options.
3. **Connection Settings** - Provide detailed information about the connection - OR -
4. **Connection String** - *Instead of providing connection setting details*, you can enter a complete connection string.
5. **Actions -** Click **Test** to test the connection, click **Done** to save any changes and return to the previous page (earlier versions may have a "Back" link at the top of the page instead of a Done button).

   Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

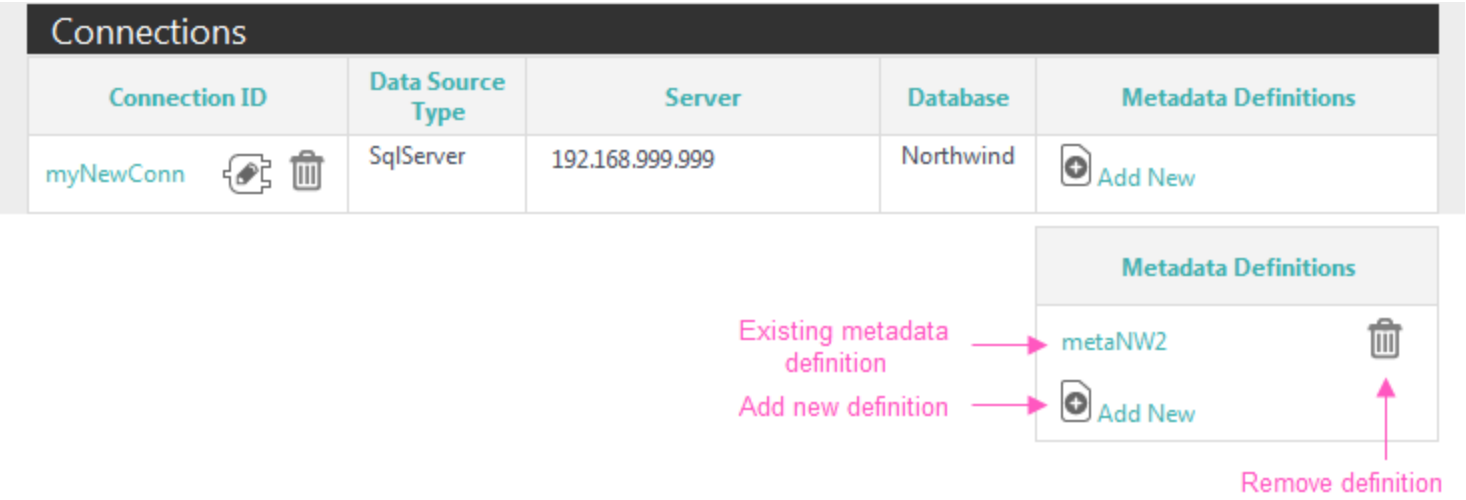
When you return to the first page, you'll see that your new connection has been added to the table of connections:



Connections can be managed using the link and icons in the first table column, as shown above.

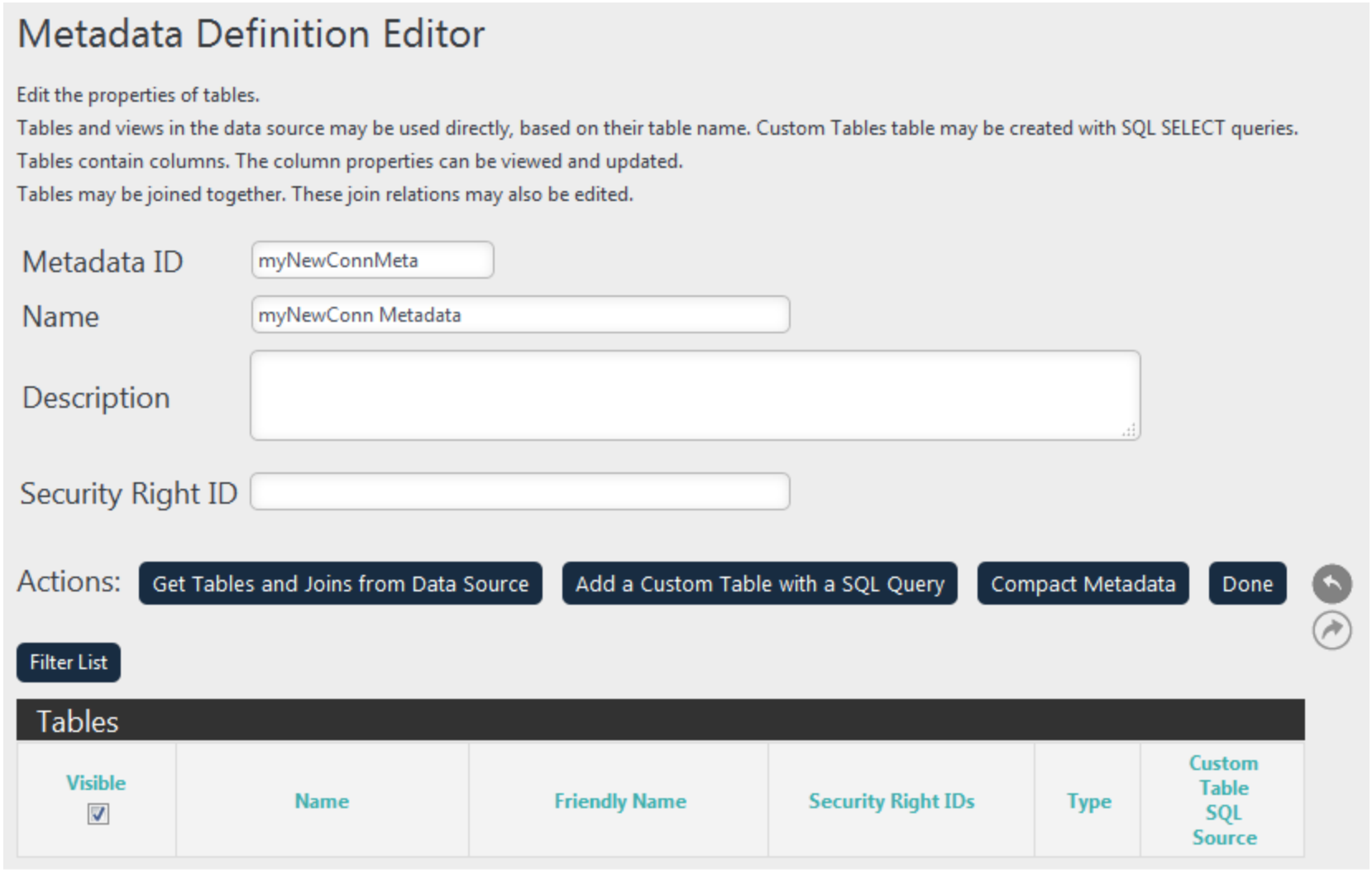
Managing Metadata Definitions

Metadata definitions are managed using the icons in the *last* table column. Additional icons become available once a definition has been created.

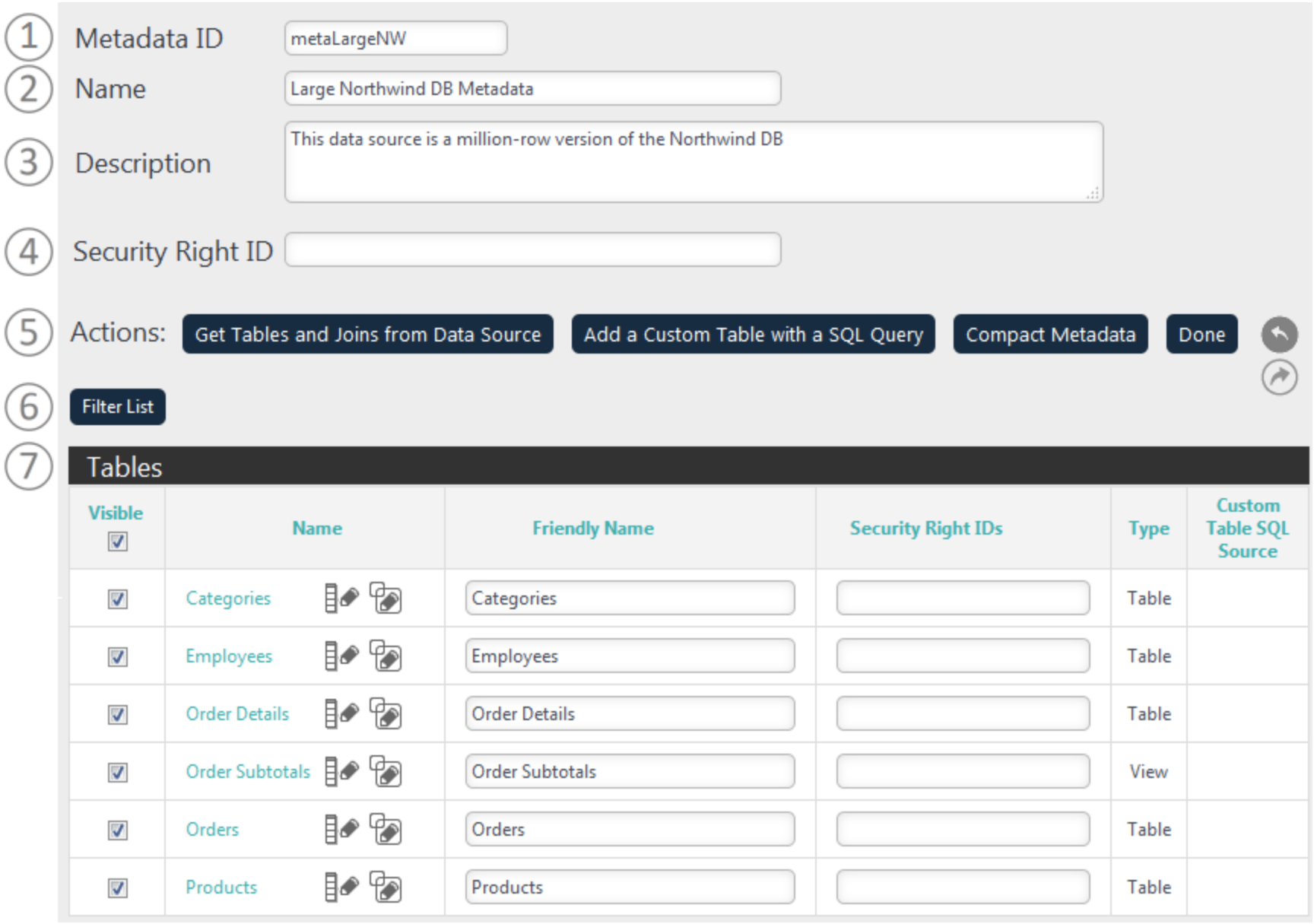


The Metadata Definition column will display the Metadata *name*, if available; otherwise it will display the Metadata *ID*.

To create a new metadata definition, click the **Add New** icon, then click **OK** in the confirmation panel that appears.



If this is new metadata definition, the page will look like the example shown above. An ID and Name will be suggested, based on the related Connection element.


1
Metadata ID
metaLargeNW
2
Name
Large Northwind DB Metadata
3
Description
This data source is a million-row version of the Northwind DB
4
Security Right ID
5
Actions:
Get Tables and Joins from Data Source
Add a Custom Table with a SQL Query
Compact Metadata
Done
6
Filter List
7
Tables
Visible
Name
Friendly Name
Security Right IDs
Type
Custom Table SQL Source
Categories
Categories
Table
Employees
Employees
Table
Order Details
Order Details
Table
Order Subtotals
Order Subtotals
View
Orders
Orders
Table
Products
Products
Table

If the metadata definition already exists, you'll see a page that looks like the one shown above. Here's an explanation of its controls:

1. **Metadata ID** - (Required) A unique element **ID** for the Metadata element that will be inserted into your _Settings definition. Do *not* include any spaces in this ID or a file extension. This value will also be used as the metadata file name, which will be created in the `_Metadata` folder with a .lgx file extension. By default, a default value will be provided associated with the related data source connection.
2. **Name** - (Required) A descriptive name. If you create multiple metadata files, this name will appear as a selection in the user interface as a "data source".
3. **Description** - More detailed information that appears in the Active Query Builder as a tooltip.
4. **Security Right ID** - A comma-separated list of one or more security Right IDs, used to control access to this Metadata definition.
5. **Actions** - Actions related to the metadata definition:

   - Click **Get Tables and Joins...** to retrieve that table and join data from the data source schema. Will not overwrite values you may have entered in Friendly Name, Security Right IDs, etc. Reverses the effects of the Compact Metadata feature and may be useful if the metadata has been altered *outside* of this tool, for example, with Logi Studio.

   - Click **Add a Custom Table**... to leave this page and create a new custom table based on a SQL query. This is discussed in more detail below.

   - Click **Compact Metadata** to physically remove, from the metadata definition, all tables in the list that do not have "Visible" checked. This makes the metadata smaller and faster to load and is useful when there are a lot of tables in the data but only a few are used. The effects of compacting can be reversed by clicking the "Get Tables and Joins..." button.

   - Click **Done** to save any changes and return to the previous page (earlier versions may have a "Back" link at the top of the

page instead of this button).

- Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

6. **Filter List** - Allows you to filter the list of tables, based on a variety of criteria, in order to work with it more easily.
7. List of **Tables** - Displays a list of tables that will be included in the metadata definition. Columns include:

The Column and Join editor pages are only available if the table has been selected as Visible.

Adding a Custom Table Based on a SQL Query

You can include custom tables, created on-the-fly from a SQL database at runtime, by clicking **Add a Custom Table with a SQL Query**. Click OK at the confirmation prompt.

Custom Table SQL Query Editor

A custom table, similar to a View, can be defined with a SQL query.
Limitations - If the SQL query contains JOINs, columns must be specifically named, no wildcards (*) allowed. Cannot include an ORDER BY clause.

1. Table Name: OrderdByID
2. SQL Query: SELECT * FROM Orders WHERE OrderID = '@Session.OrderID~'

Optional Session Variables in SQL Queries

Session tokens may be included in the SQL Query's WHERE clause and will be resolved at runtime.
For example: "SELECT * FROM Orders WHERE EmployeeID = '@Session.EmployeeID~' ".

Working Values for @Session Tokens When @Session tokens are included in the SQL Query, supply working values to be used when testing the query and getting the columns. Add names and values for each @Session token. The tokens then get replaced by the working values.

3.

Remove	Session Variable Name	Working Value
🗑	OrderdByID	10248

Add another @Session Token Working Value

4. Actions: Test SQL Query | Get Columns from the Data Source | Done

The Custom Table SQL Query Editor page will appear, as shown above. If you're editing an existing custom table, the controls will be filled-in with the current definition. The page's controls are:

1. **Table Name** - (Required) A unique name for the custom table.
2. **SQL Query**- (Required) The SQL query that will retrieve data for your custom table. If your query includes Joins, you may not use wildcards as columns must be specifically named. You may not include an ORDER BY clause in any query. Session tokens may be used in a WHERE clause and will be resolved at runtime.

   You may not use an EXEC statement to call a Stored Procedure in order to create the custom table.
3. **Session Variables** - Session tokens can be used in your SQL query, and these controls allow you to provide "working" values for them that will be used when the WMB tests the query and retrieves column information. Click the "Add another..." link to add additional session tokens and values.
4. **Actions** - Actions related to the custom table definition:

   - Click **Test SQL Query** to execute the query against the data source and display the first 100 rows returned. This will not work if a token is used in the query, as it won't exist. You may care to temporarily hard-code a value in place of the token in order to test it.

   - Click **Get Columns from the Data Source** to execute the query against the data source and display the first 100 rows returned AND add the columns to the metadata definition. The same limitations apply regarding tokens as with the Test SQL Query feature.

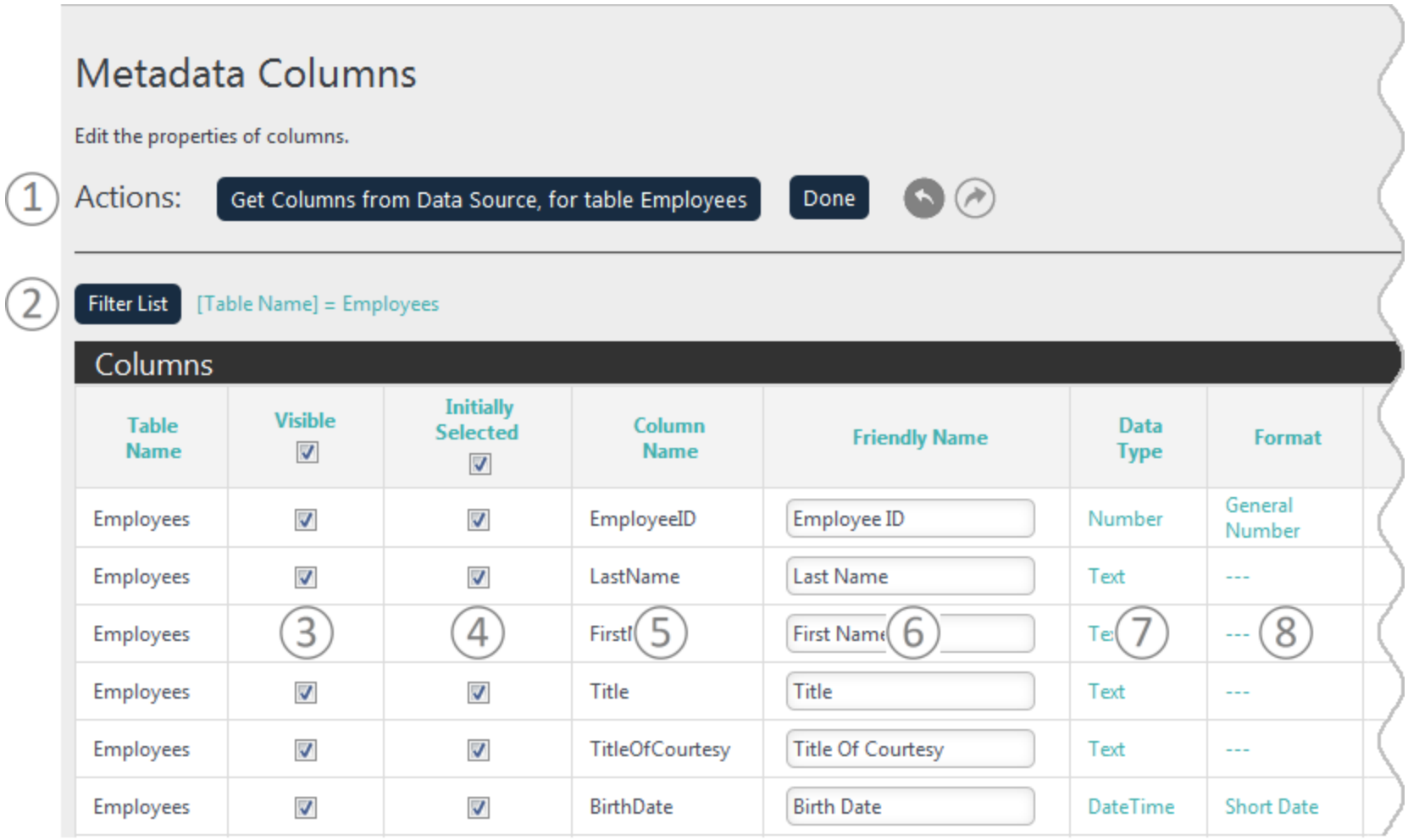
   - Click **Done** to save any changes and return to the previous page(earlier versions may have a "Back" link at the top of the page instead of this button).

   - Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

New custom tables will appear in the table list, with some additional icons, as shown above. The table Type will be shown as "Custom" and the (possibly truncated) SQL Query code will also appear. The custom table can then be configured in the same manner as any regular table, with the configuration of columns, friendly name, and Security Right IDs.

Using the Column Properties Editor Page

By default, all columns in visible tables are included in the metadata, using standard properties. If you want to specify their properties more exactly, click the **Edit Columns** icon for the desired table and the Column Properties Editor page will be displayed:



The Column Properties Editor is very wide, so we're going to show it to you in two horizontal parts. The left part is shown above and includes the following details and controls:

1. **Actions** - Actions related to the column properties:

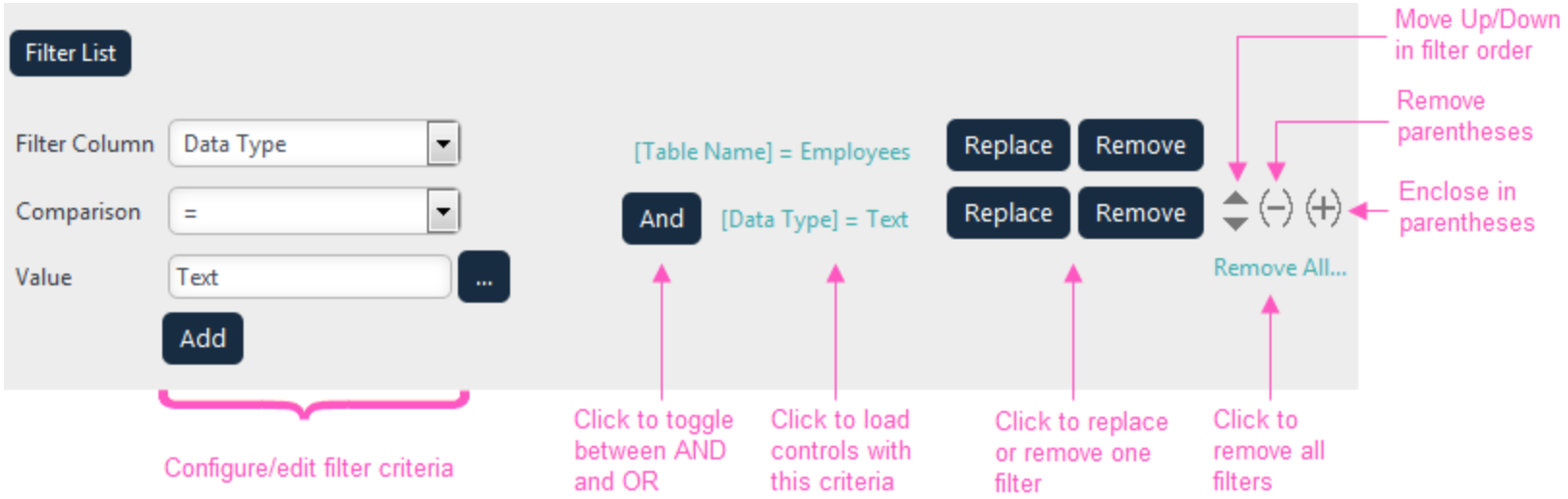
   - Click **Get Columns from Data Source** to retrieve the schema information for *all* of the columns for this table from the

data source. This may be useful if the metadata has been altered *outside* of this tool, for example, with Logi Studio.

- Click **Done** to save any changes and return to the previous page(earlier versions may have a "Back" link at the top of the page instead of this button).

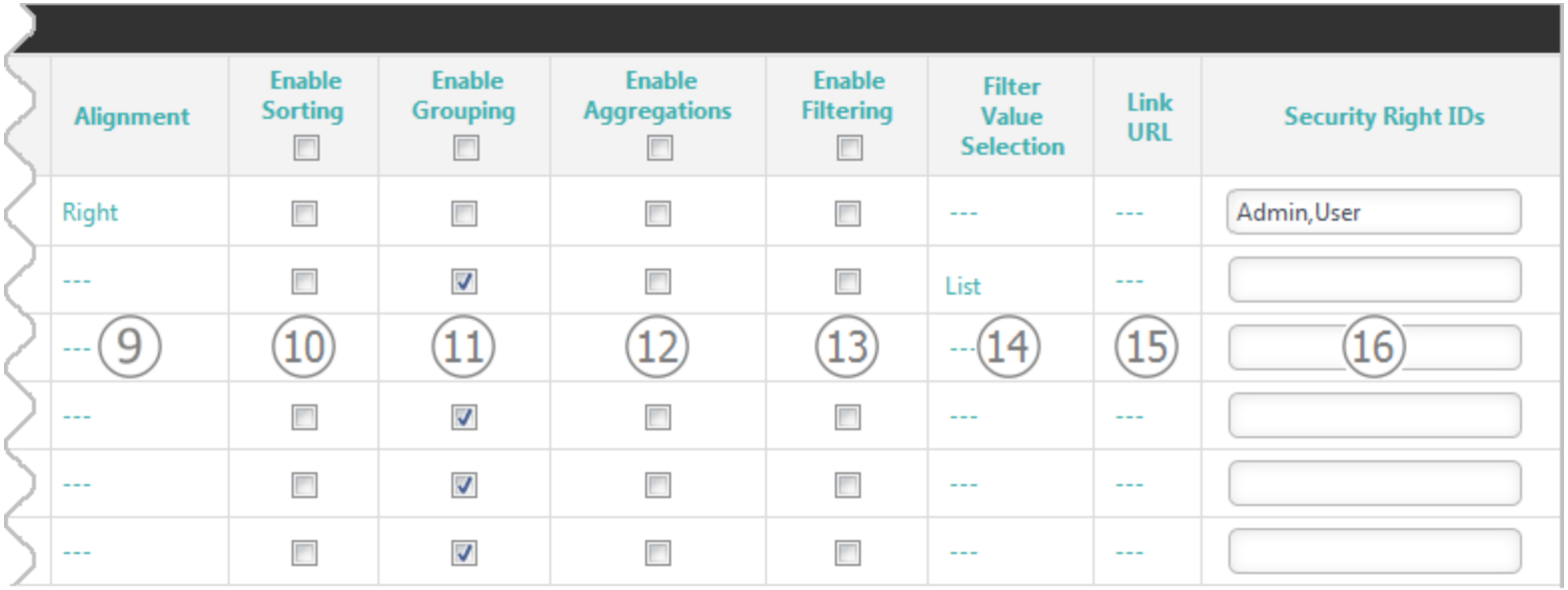
- Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

2. **Filter List** - Allows you to filter the list of columns, based on a variety of criteria, in order to work with it more easily. The list is initially filtered on the name of the table whose icon you clicked in the list of tables on the previous page. The filter controls work like this:



3. **Visible** - Check the box for each column to be included in the metadata. Check the box in the column header to select/unselect all columns at once.

4. **Initially Selected** - Check the box for each column to be initially selected for viewing in the Data Table in the target visualization element (i.e. Analysis Grid). Check the box in the column header to select/unselect all columns at once.
5. **Column Name** - The actual column name in the data source.
6. **Friendly Name** - (Required) The "user-friendly name" for each column that will appear in the user interface. Edit this to provide end-users with a more understandable or relevant column name.
7. **Data Type** - Specifies a data type for each column. The data type specified here affects the default column display format. Click this link to select the data type in a pop-up panel. Options include *Boolean*, *Date*, *DateTime*, *Number*, and *Text*.
8. **Format** - Specifies the display format for each column. The default format is based on the column data type. Click this link to select the format in a pop-up panel and 20 options are available. Leave this blank ("---") to use the data type default.



9. **Alignment** - Specifies an alignment (*Left*, *Center*, *Right*) for data displayed in each column. If nothing is specified, a default alignment will be used based on the data type.
10. **Enable Sorting** - Specifies whether this column can be sorted; check the box to allow it.
11. **Enable Grouping** - Specifies whether data can be grouped on this column; check the box to allow it.
12. **Enable Aggregations** - Specifies whether data in this column can be aggregated; check the box to allow it.
13. **Enable Filtering** - Specifies whether this column can be used in a filter; check the box to allow it.
14. **Filter Value Selection** - Specifies the option that will appear for this column with filtering controls: nothing, a *List* of columns, or a pop-up *Calendar*.
15. **Link URL** - Specifies that the data in this column will be a link, with a dynamic URL associated with it. Click to open a pop-up panel where you can enter the URL code and select a destination window (*New*, *Parent*, or *Top*).

     The data will *not* be rendered as a live link in any tables or visualizations created using the Thinkspace.

    Use the following format for your URL entry, substituting your application, report name, request variable, table, and column names. The special token @Data.*TableName_ColumnName*~ will be replaced with actual data values at runtime. You can also run JavaScript by beginning the URL with "javascript:". Examples:

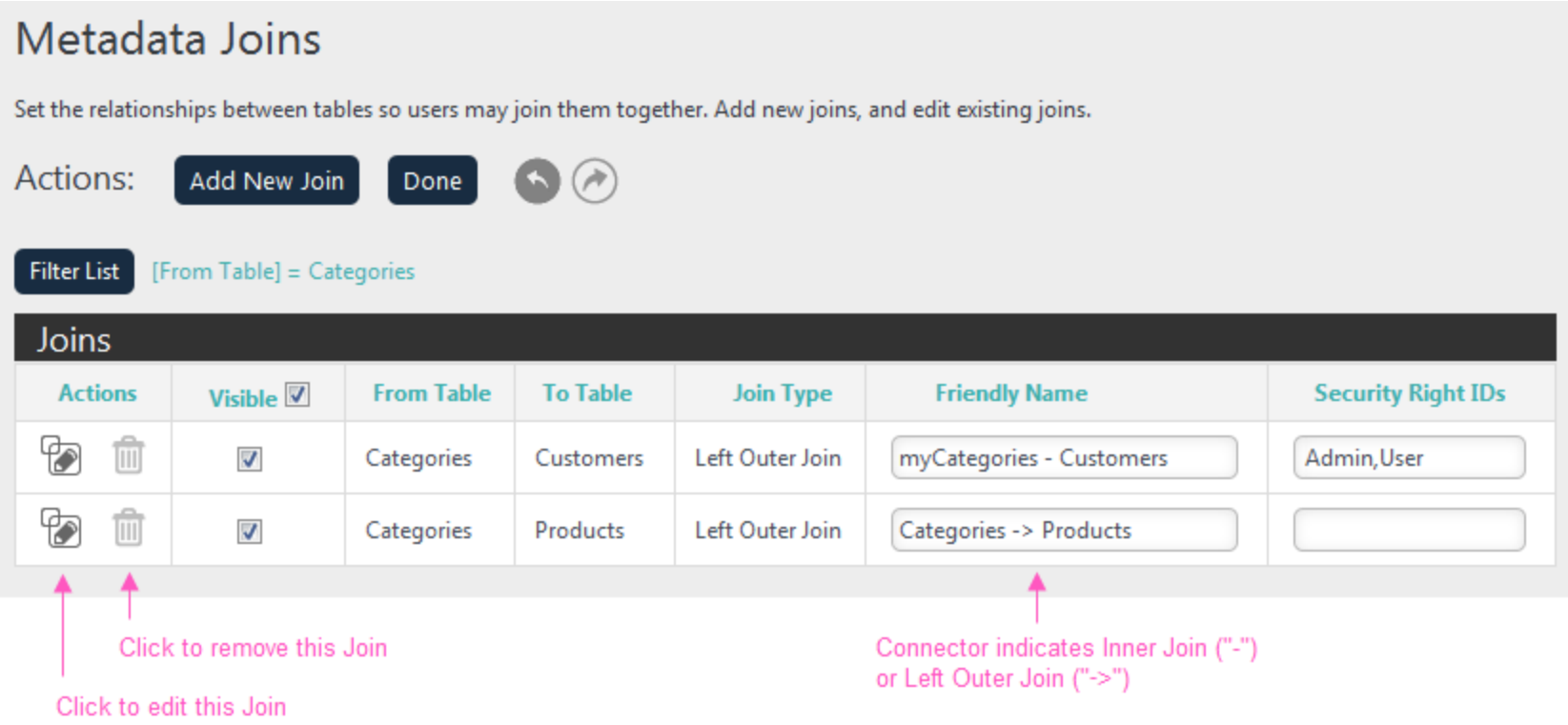
    ```
    http://myServer/myApp/rdPage.aspx?rdReport=myOrdersReport&OrderID=@Data.myOrdersTable_OrderID~
    http://www.bing.com/search?q=@Data.Categories_CategoryName~
    javascript:myScript('@Data.Categories_CategoryName~');
    ```
16. **Security Right IDs** - Select, or enter a comma-separated list of, Security Right IDs which will control access to the column, when Logi Security is in use.

Using the Join Editor Page

This page allows you to manage the Joins, or relationships, between tables. The table whose icon you clicked in the list of tables is considered the "From" table. Another table you select later is considered the "To" table.

The WMB supports two types of Joins: an *Inner Join*, which returns all rows from both tables, as long as there is a match on the "From" column and the "To" column, and a *Left Outer Join*, which returns all rows from the "From" table, even if there are no matches in the "To" table, with NULL values in the right side when there is no match.

To manage Joins, click the **Edit Joins** icon  for one of the tables:



If you have existing Joins defined, you'll see them in the list of Joins, as shown above.

Actions - Actions related to the Join definitions:

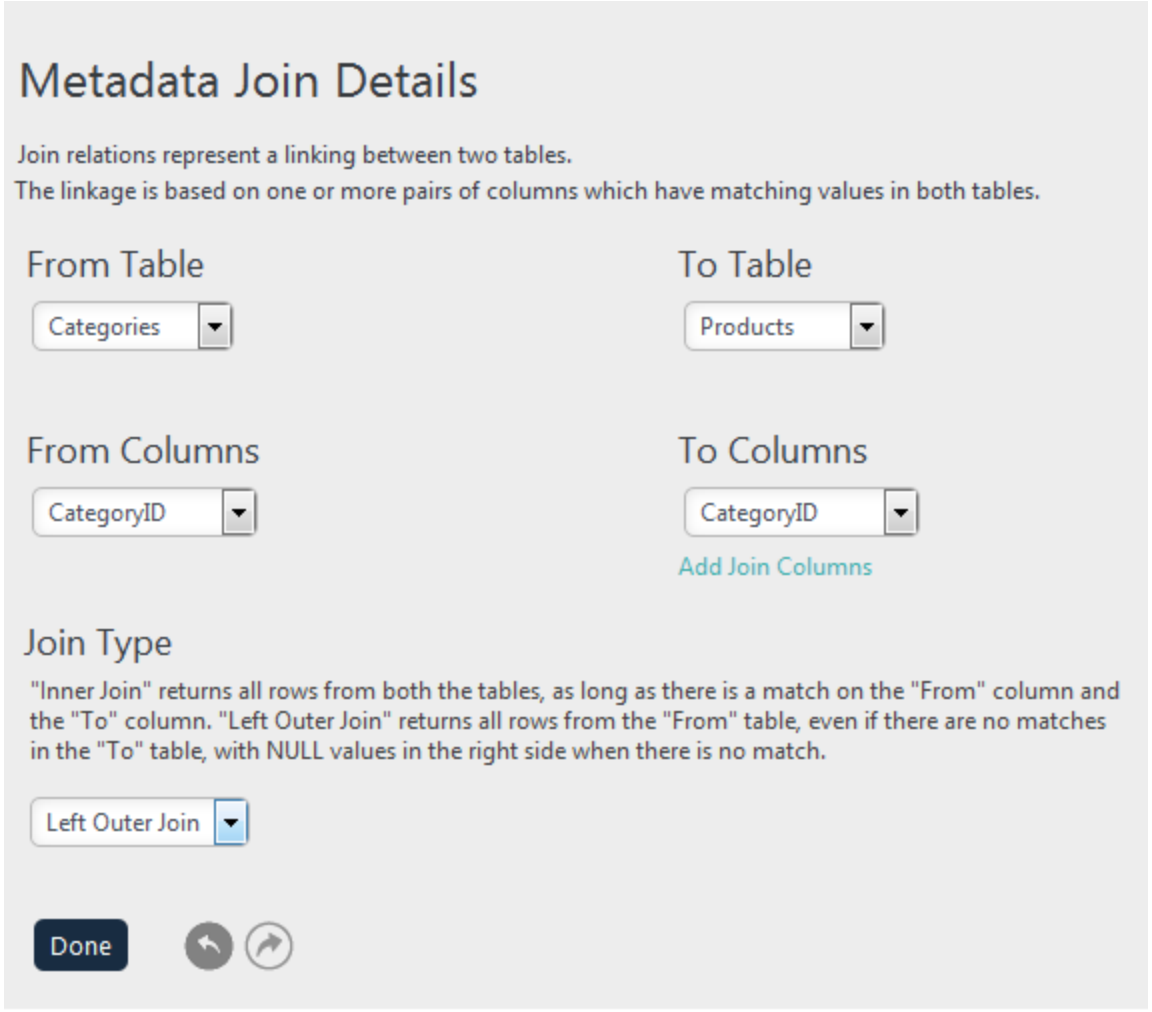
- Click **Add New Join** to define a new Join involving the current table.
- Click **Done** to save any changes and return to the previous page(earlier versions may have a "Back" link at the top of the page instead of this button).
- Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

Filter List - Allows you to filter the list of Joins, based on a variety of criteria, in order to work with it more easily. See the previous section for a discussion of the filter controls.

Click the icons shown above to edit or remove a specific Join.

The **Friendly Names** that the WMB suggests when a Join is created include a table and column name connected by a symbol. The symbol indicates whether it's an *Inner Join* ("-") or a *Left Outer Join* ("->") and this is informative when using the Active Query Builder later to select data for analysis.

If you click Add New Join, and then OK in the confirmation panel that appears, the following page will be displayed:



To create a Join, you simply select the tables and columns to be related and the desired Join type, from the selection lists. You can join additional columns by clicking **Add Join Columns**.

Click **Done** to save any changes and return to the previous page (earlier versions may have a "Back" link at the top of the page instead of this button).

Click the **Undo/Redo** icons to reverse and retry changes, if automatic bookmarks are being used.

The Metadata File

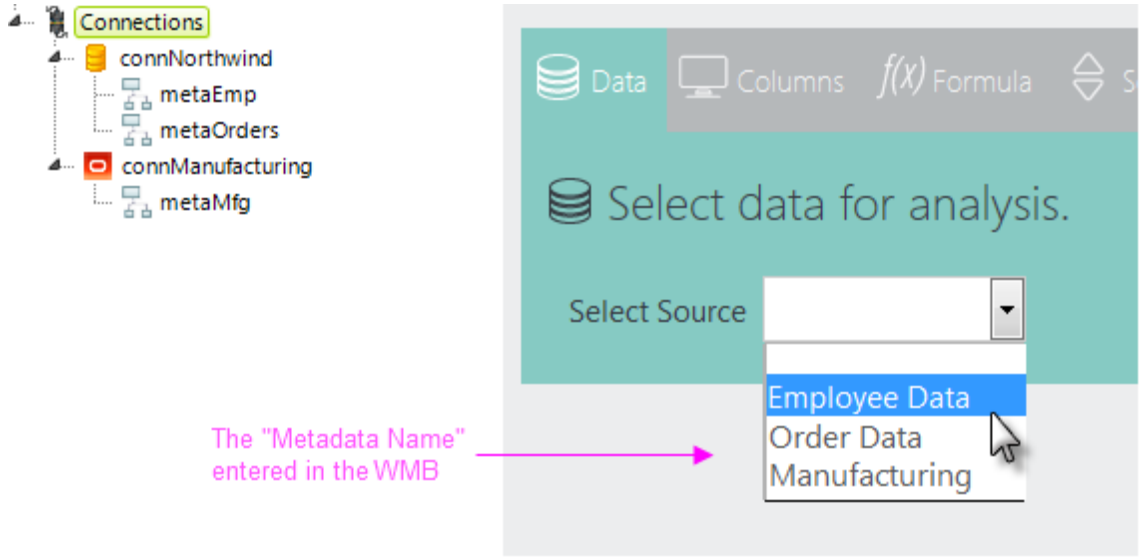
The WMB has been automatically saving any changes you've made to the metadata file:

```
<?xml version="1.0" encoding="utf-8"?>
<Metadata MetadataBuilder="Logi Metadata Builder" BuilderVersion="11.3.001">
  <Table TableName="Order Details" FriendlyName="Order Details" TableType="U
    <Column ColumnName="OrderID" FriendlyName="Order ID" DataType="Number" F
    <Column ColumnName="ProductID" FriendlyName="Product ID" DataType="Numbe
    <Column ColumnName="UnitPrice" FriendlyName="Unit Price" DataType="Numbe
    <Column ColumnName="Quantity" FriendlyName="Quantity" DataType="Number"
    <Column ColumnName="Discount" FriendlyName="Discount" DataType="Number"
  </Table>
  <Table TableName="Orders" FriendlyName="Orders" TableType="U" HideTable="F
    <Column ColumnName="OrderID" FriendlyName="Order ID" DataType="Number" F
    <Column ColumnName="CustomerID" FriendlyName="Customer ID" DataType="Tex
    <Column ColumnName="EmployeeID" FriendlyName="Employee ID" DataType="Num
    <Column ColumnName="OrderDate" FriendlyName="Order Date" DataType="DateT
    <Column ColumnName="RequiredDate" FriendlyName="Required Date" DataType=
    <Column ColumnName="ShippedDate" FriendlyName="Shipped Date" DataType="D
    <Column ColumnName="ShipVia" FriendlyName="Ship Via" DataType="Number" F
    <Column ColumnName="Freight" FriendlyName="Freight" DataType="Number" Fo
    <Column ColumnName="ShipName" FriendlyName="Ship Name" DataType="Text" F
    <Column ColumnName="ShipAddress" FriendlyName="Ship Address" DataType="T
    <Column ColumnName="ShipCity" FriendlyName="Ship City" DataType="Text" F
    <Column ColumnName="ShipRegion" FriendlyName="Ship Region" DataType="Tex
    <Column ColumnName="ShipPostalCode" FriendlyName="Ship Postal Code" Data
    <Column ColumnName="ShipCountry" FriendlyName="Ship Country" DataType="T
    <Column ColumnName="Invoiced" FriendlyName="Invoiced" DataType="Text" Fo
  </Table>
  <JoinRelation FromTableName="Orders" ToTableName="Order Details" Relation=
4a16-adc4-98b8c2156007" SecurityRightID="">
    <JoinRelationDetails FromColumnName="OrderID" ToColumnName="OrderID" />
  </JoinRelation>
</Metadata>
```

An example of the contents of a metadata file is shown above. It's an XML file that describes all of the data, relationships, and other details you specified in the WMB.

Creating Multiple Metadata Files

As mentioned earlier, you can create multiple metadata files in order to provide multiple datasets to users.



As shown in the example above, the WMB is capable of inserting multiple child Metadata elements beneath a single Connection element. Multiple Connection elements, each with their own child Metadata elements, can also be added. The result, in the Active Query Builder, is a combined list of datasets for the user to select for analysis.

For information about using metadata with the Active Query Builder, see *The Analysis Grid for Developers*.

Web Services

This topic provides an overview of **web services** and their use with Logi applications.

The following sections are covered in this topic:

- About Web Services
- Connectivity
- Working with Data

About Web Services

A **Web Service** is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network". Web services are frequently just web-based APIs that can be accessed over a network and executed on a remote system hosting the requested services.

In common usage, "web service" refers to clients and servers that communicate using XML or JSON messages, following the **SOAP** or **REST** protocol standards. Logi Info uses XML as its underlying technology and is therefore well-suited to work with web services.

Logi Studio makes it easy to work with web services by doing some of the required legwork for you. For example, web services often provide a public Web Services Description Language (WSDL) file describing its methods. Logi Studio's wizards can read this file and the API methods and their parameters to you.

You may need to create an account in order to access a web service and some services may require that commercial users pay for the use of their service. Those activities and arrangements are beyond the scope of this topic.

To interact with REST-style web services from within a Process task, see *Process Tasks*.

Connectivity

In a Logi application, the same basic technique that's used for connecting to a database is used to connect to a web service: a Connection element. Special Connection elements are added to the _Settings definition to provide connectivity.

Connection.Web Service is used to connect to a SOAP-style web service and has an optional child element, **Connector Property Parameters**, which can be used handle special authentication and proxy settings for the connection.

Connection.REST is used to connect to a REST-style web service and has an optional child element, **Request Header**, which can be used (in multiples, if needed) to add special attributes and values to the request header sent in an HTTP communication with the service.

- If you're connecting to *your own* web service using Connection.Web Service, the WSDL document specified in this connection element's required **WSDL URL** attribute must be "valid", meaning that it can contain no errors that would cause it to fail to compile. A document that's readable, but that will not compile, is *invalid* and will cause the connection to fail.
- If you're trying to communicate with a web service that requires the **TLS 1.1** or 1.2 protocol, you will need to use Logi Info v12.2-SP4 or later (earlier versions only support TLS 1.0). In addition, Info Java applications must use Oracle JDK 1.8 or OpenJDK 8 to make the protocol work.

Once defined, this connection is then referenced in other report definitions that wish to use the web service. The process is quick and easy.

Working with Data

The same basic technique that's used to retrieve data from a database is used to retrieve data from a web service. You use one of these special datalayer elements:

- *DataLayer.Web Service*
- *DataLayer.REST*
- *DataLayer.JSON*
- *DataLayer.XML*

which in combination with the Connection mentioned earlier, receives the data retrieved from the web service. All of them have special child elements that can be used to pass parameters to the web service.

 The last three are capable of retrieving the data by themselves, without the use of a Connection element, in certain circumstances.

All datalayers expect to receive data beginning with the relevant nodes. However, this may not always be the case: results may have additional parent nodes before the data. To deal with this, the datalayers have an **XPath** attribute where XPath queries can be entered for the purpose of selecting the desired data nodes for the datalayer.

Once data from the web service has been retrieved into a datalayer, then all of the usual Logi data handling (filtering, aggregating, etc.) and presentation elements can be applied to it.

XML vs JSON

The two primary web service return data formats are XML and JSON. XML long been a popular way to structure data using a familiar markup language. Despite its many advantages, one drawback of XML is its size: it contains many characters strictly related to

formatting. When downloading data to a mobile device, it would be ideal to just obtain the relevant content instead of data related to content formatting. This is where JSON comes in.

JavaScript Object Notation (JSON) is a text-based data interchange format derived from the JavaScript scripting language. It is formatted as key-value pairs and is said to have lower "overhead" than XML because it focuses more on content and less on formatting. Also, many popular open source libraries, such as JQuery, make extensive use of JSON data.

Regardless of the data format returned by the web service you want to work with, Logi Info includes components for creating useful analyses.

Hadoop

The Apache **Hadoop** software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

Hadoop drivers were deprecated from Info Java. The following libraries were removed: `hadoop-core-0.20.2.jar`, `hive-exec-0.13.0.jar`, `hive-jdbc-0.13.0.jar`, `hive-service-0.13.0.jar`, `libfb303-0.9.1.jar`, and `libthrift-0.9.1.jar`.

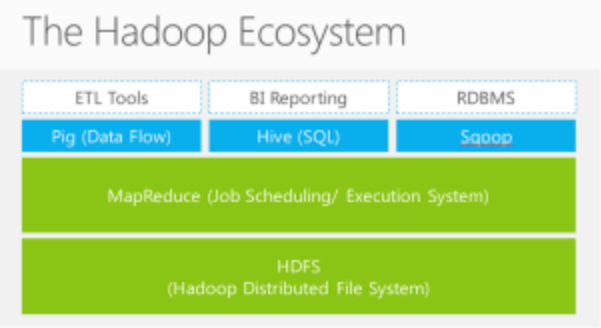
The following topics discuss how to work with Hadoop implementations:

- Connecting to a Hadoop Datasource
- Retrieving Data
- Cloudera and Kerberos Authentication

General information about Hadoop can be found at the official website.

About Hadoop

Logi Analytics has strategic partnerships with the industry's Big Data technology leaders for analytical and Hadoop data stores: HP Vertica, Amazon Redshift, ParStream, Hortonworks, and Cloudera.



Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect and handle failures at the application layer, delivering a highly-available service on top of a cluster of computers.

Logi Info accesses data in real-time, through an ODBC connector in "Hive", a Hadoop component which facilitates querying and managing large datasets residing in distributed storage. Hive projects structure onto this data and queries the data using a SQL-like language called **HiveQL**. At the same time this language also allows traditional map/reduce programmers to plug in custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Logi Studio's SQL Query Builder tool works well with HiveQL, to help build managed reports quickly and efficiently.

This topic presents techniques for connecting Logi Info applications to Hadoop implementations, such as Cloudera CDH4 and Hortonworks, and discusses the details of setting up Cloudera Kerberos authentication.

Connecting to Hadoop

For a .NET Logi application, use the **Connection.ODBC** element to connect your Logi application to a the datasource.

For a Java Logi application, use the **Connection.JDBC** element. Logi Info supports Apache Hive 0.13.0 with Hortonworks 2.1 or Cloudera 5.0 Impala, and include supporting drivers and libraries for them.

Hadoop drivers were deprecated from Info Java. The following libraries were removed: `hadoop-core-0.20.2.jar`, `hive-exec-0.13.0.jar`, `hive-jdbc-0.13.0.jar`, `hive-service-0.13.0.jar`, `libfb303-0.9.1.jar`, and `libthrift-0.9.1.jar`.

Special ODBC Drivers Required

For .NET applications, depending on which Hadoop implementation you're connecting to, you'll need to *download and install* either the Cloudera ODBC Driver for Impalaor the Hortonworks Hive ODBC Driver. Select the proper driver version and Windows version (32- or 64-bit) for your application (a 32-bit driver is required for a 32-bit Logi app, and a 64-bit driver for a 64-bit app). You can install 32- and/or 64-bit drivers on a 64-bit machine.

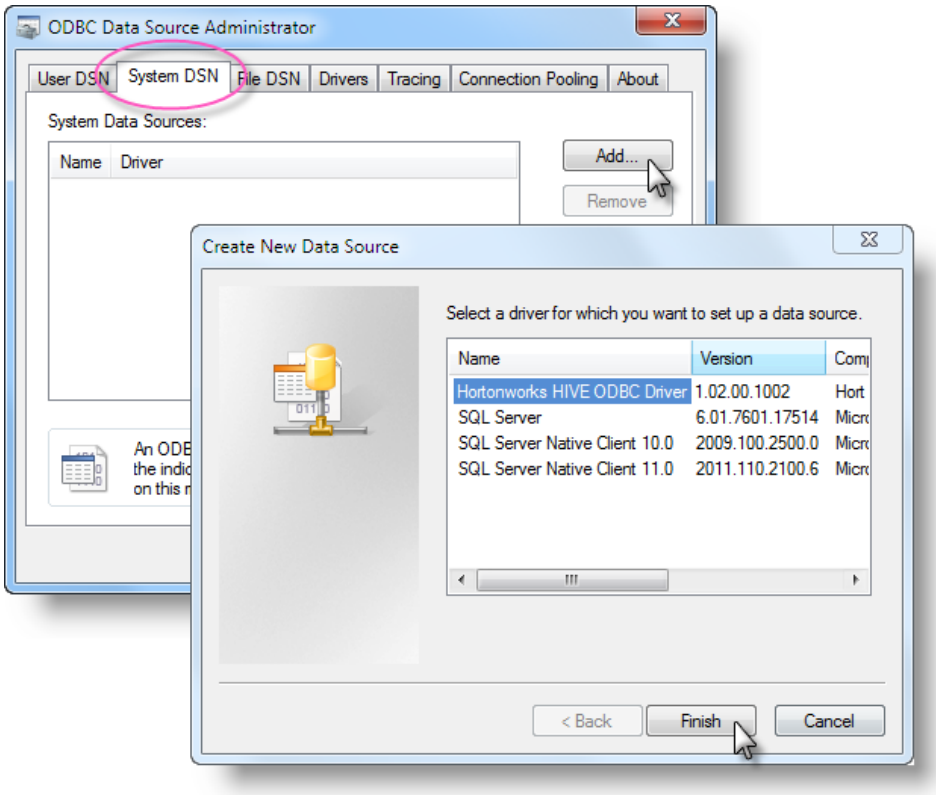
The links provided point to web pages that include the drivers and installation guides. In general, all the default values can be used. After installation, **restart** the computer.

ODBC Data Source Configuration

For .NET applications, once you have a driver installed, you'll need to configure a ODBC data source to use it. This is done using the Windows **ODBC Data Source Administrator** utility.

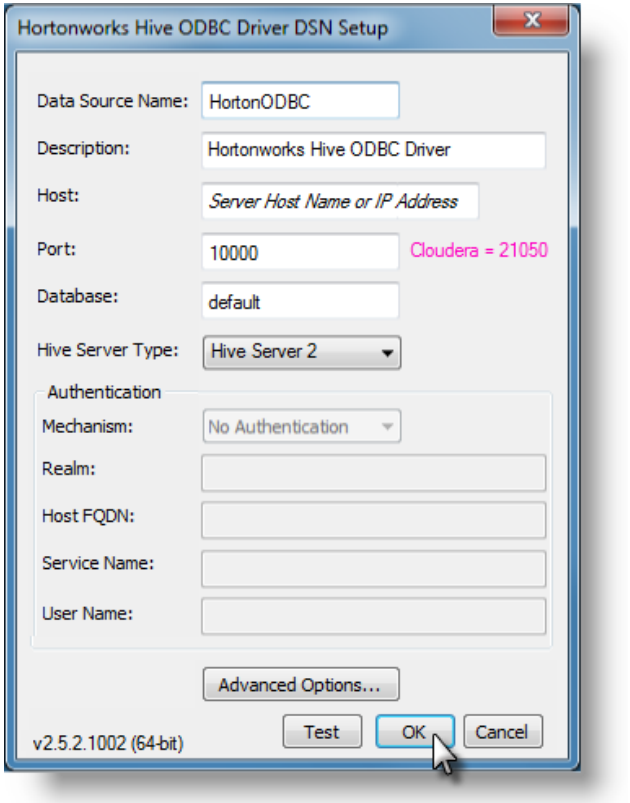
On Windows platforms, this tool is typically launched from Start Menu → Administrative Tools → Data Sources, or Start Menu → Control Panel → Administrative Tools → Data Sources. These shortcuts launch the utility that matches the OS, i.e. on a 32-bit OS they launch the 32-bit utility, on a 64-bit OS, the 64-bit utility.

If you want to configure a 32-bit data source on a 64-bit machine, you *must* use the *32-bit version* of the Data Source Administrator. This is the source of many confusing problems when what appears to be a perfectly configured ODBC DSN does not work because it is loading the wrong kind of driver. You can't access the 32-bit Data Source Administrator from the Start Menu or Control Panel in 64-bit Windows. Instead, you must manually launch `C:\WINDOWS\SysWOW64\odbcad32.exe`.



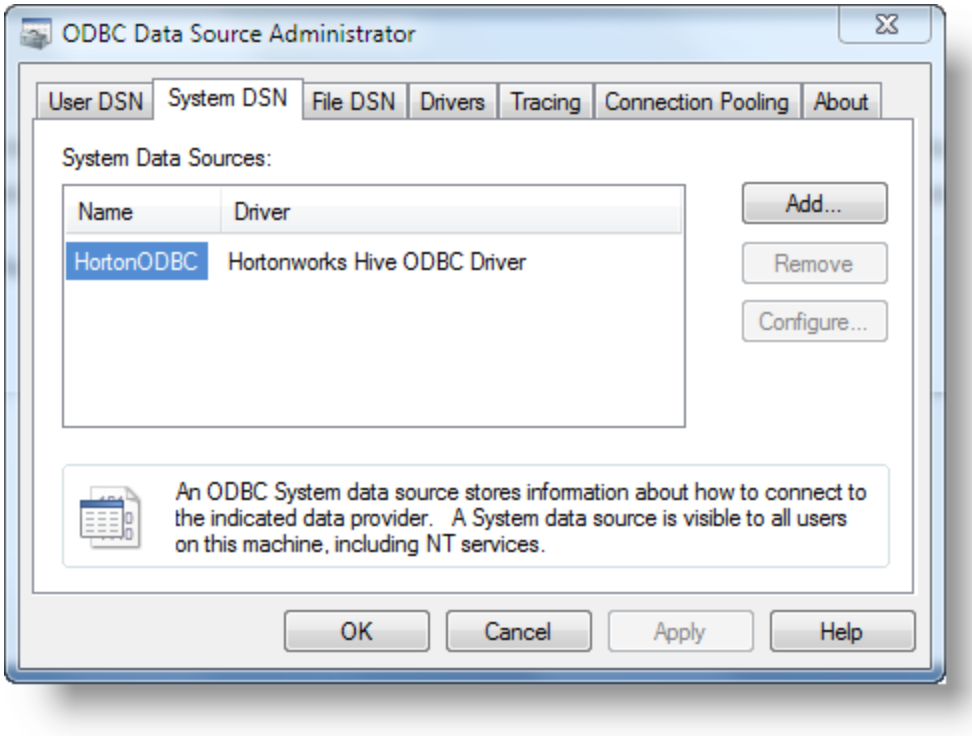
In the following examples, the Hortonworks driver will be shown, but the information also generally applies to the Cloudera driver. In the Data Source Administrator, shown above, select the **System DSN** tab and click **Add**. Select your new driver from the list and click **Finish**.

This *must* be created as a System DSN, *not* a User DSN. A User DSN will not be found by the web server and a "Data source name not found" error will occur.



The Driver Setup dialog box, shown above, will be displayed. Fill-in the information as suggested above, making appropriate changes for the Cloudera driver. Detailed information about the purpose of each field is available in the *Cloudera*and *HortonworksDSN Installation Guides*. Authentication configuration is discussed in "Cloudera and Kerberos Authentication" on page 180.

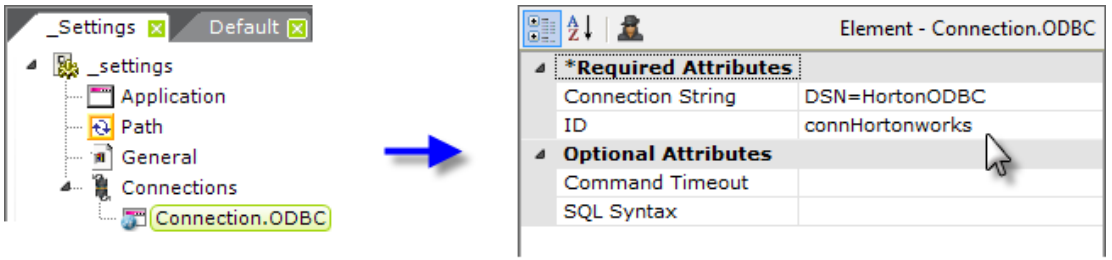
Once you've entered appropriate data, click **Test** to validate the connection. Click **OK** to save the DSN setup.



Your new data source should appear in the System DSN list, as shown above. Click **OK** to exit the utility. Now you're ready to use the driver.

Making the Connection

For .NET applications, use a **Connection.ODBC** element to make the connection. For a Java applications, use the **Connection.JDBC** element.

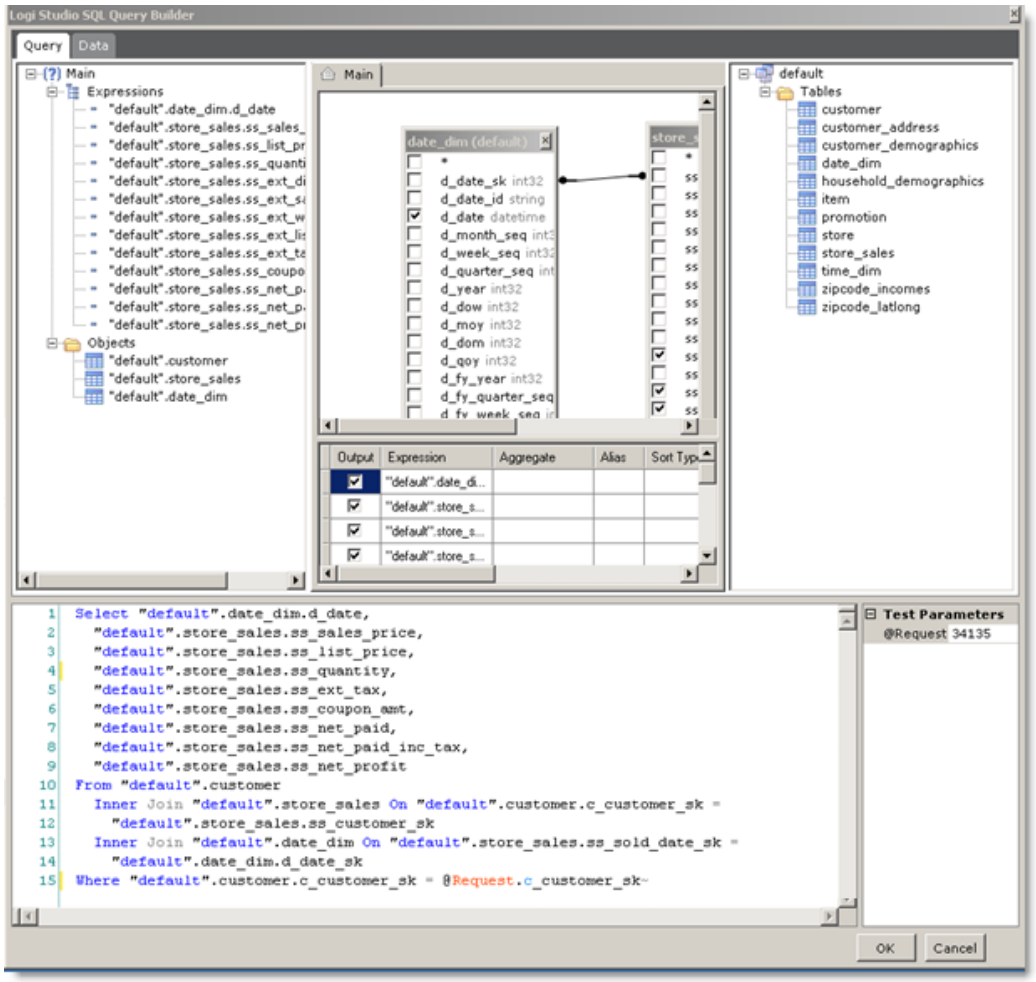


The example above shows a **Connection.ODBC** element in the **_Settings** definition, beneath the Connections element. Its Connection String attribute is then set to use the new ODBC datasource, by DSN name.

Retrieving Data

You can use **DataLayer.SQL** to retrieve data from Hadoop, and the Studio's SQL Query Builder tool can help you formulate a query. It can be invoked from the datalayer's **Source** attribute.

- Hadoop drivers were deprecated from Info Java.
- DataLayer.Active SQL is not currently available for use with this datasource.




```
Select "default".date_dim.d_date,
  "default".store_sales.ss_sales_price,
  "default".store_sales.ss_list_price,
  "default".store_sales.ss_quantity,
  "default".store_sales.ss_ext_tax,
  "default".store_sales.ss_coupon_amt,
  "default".store_sales.ss_net_paid,
  "default".store_sales.ss_net_paid_inc_tax,
  "default".store_sales.ss_net_profit
From "default".customer
  Inner Join "default".store_sales On "default".customer.c_customer_sk =
    "default".store_sales.ss_customer_sk
  Inner Join "default".date_dim On "default".store_sales.ss_sold_date_sk =
    "default".date_dim.d_date_sk
Where "default".customer.c_customer_sk = @Request.c_customer_sk~
```

As mentioned earlier, Hadoop uses **HiveQL** which is based on standard SQL syntax but has a number of very important differences that you should be aware of as you create your queries. Please refer to the *Hive Language Manual* for more information. After the data is retrieved, you may condition, filter, and shape it just as you would any other datalayer data.

Another Studio tool, the **Database Browser**, is *not* compatible with HiveQL.

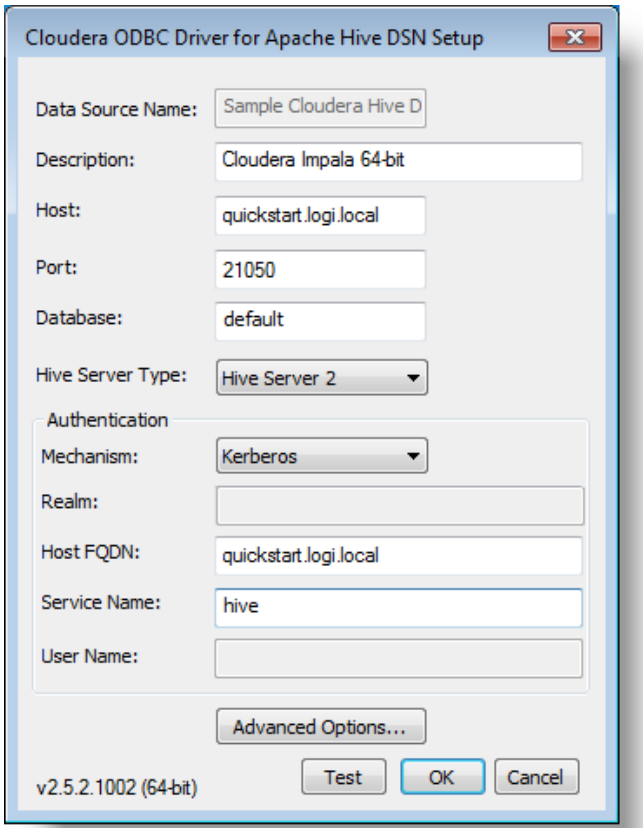
Once data has been retrieved into the datalayer, it can be manipulated using any of the elements typically used to filter, aggregate, and condition data. For more information on how to manipulate data, see *Manipulating the Data*.

Cloudera and Kerberos Authentication

Logi Info has been certified with Cloudera's CDH4 using Kerberos as a method to authenticate database access. *Hortonworks is not certified at this time*.

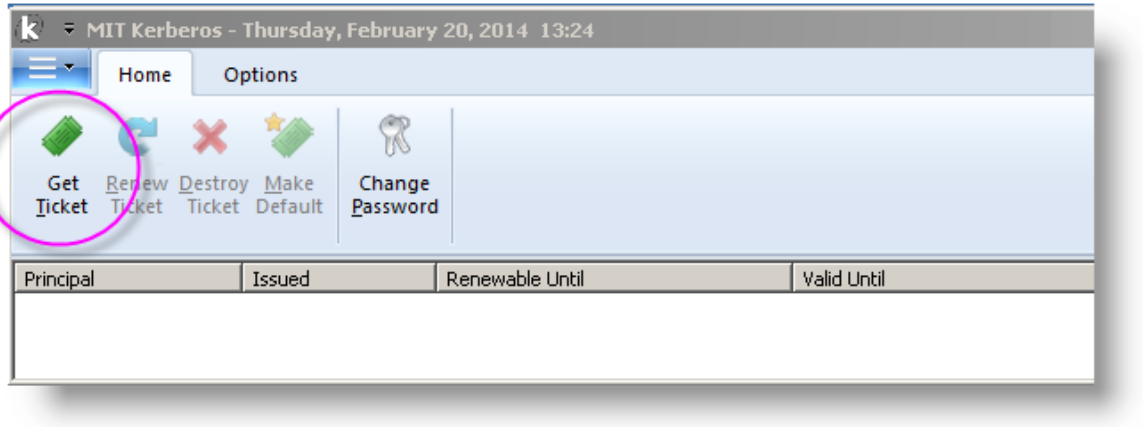
 Hadoop drivers were deprecated from Info Java. The following libraries were removed: `hadoop-core-0.20.2.jar`, `hive-exec-0.13.0.jar`, `hive-jdbc-0.13.0.jar`, `hive-service-0.13.0.jar`, `libfb303-0.9.1.jar`, and `libthrift-0.9.1.jar`.

Kerberos security should be enabled at the cluster server by following the instructionson the Cloudera web site. MIT's `kfw-4.0.1-amd64.msi`must be installed on the Logi application web server, and then the file `krb5.conf`should copied from the Kerberos server to the web server. Save it to the following folder and *change its file extension* as shown: `C:\ProgramData\MIT\Kerberos5\krb5.ini`.

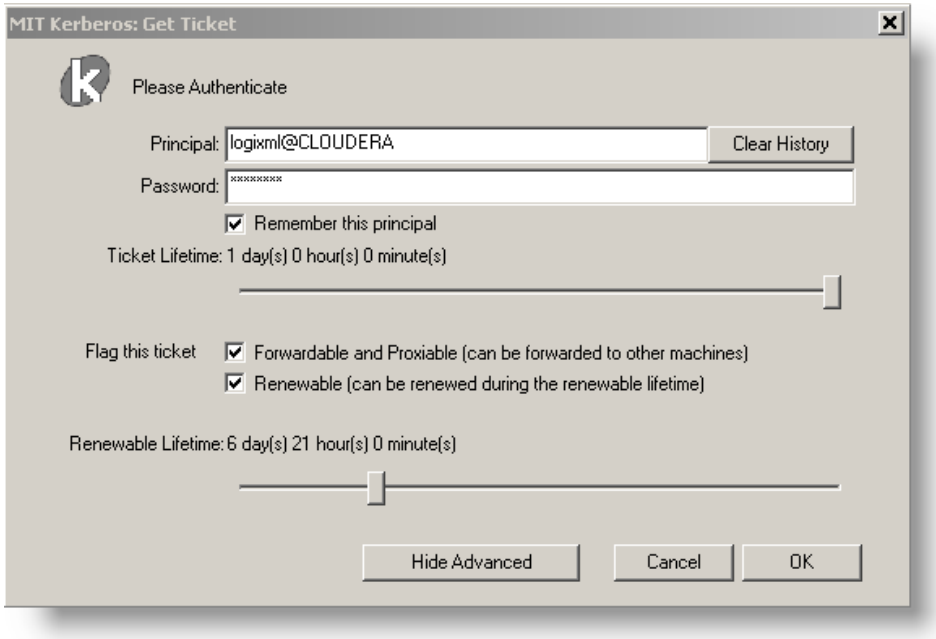


A typical DSN setup is shown above. Kerberos will *not* work with IP addresses in place of a **Host** name.

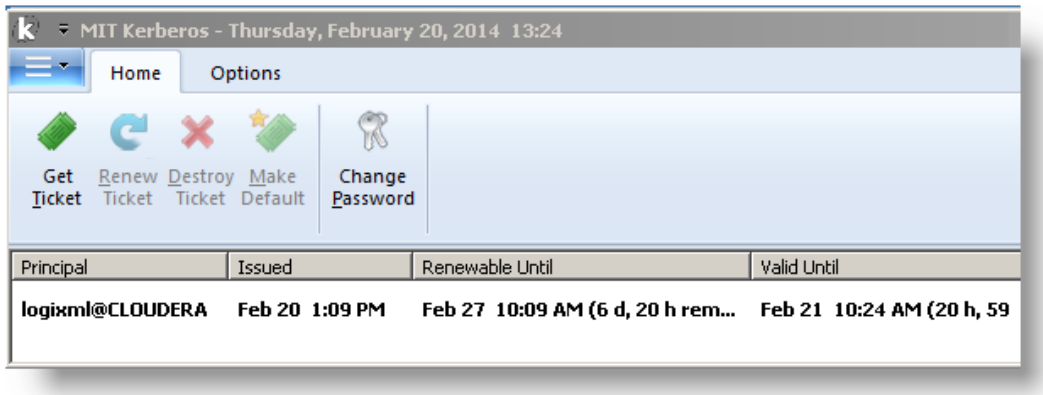
For this example, a Hadoop account was created for "logixml" and added to the Kerberos domain ("CLOUDERA"). Then this account is used in the Impala configuration to allow access as the Kerberos principal user.



The MIT Kerberos Ticket Manager utility, which is part of the `kfw-4.0.1-amd64.msi` installation and shown above, is used to get a Kerberos ticket.



On the server, the MIT Kerberos Get Ticket application is used to obtain the correct credentials from the Kerberos domain controller.



Once the credentials are obtained, they appear in the Ticket Manager, as shown above.

Working with HP Vertica

The column-oriented **HP Vertica** database is designed to manage large, fast-growing volumes of data.

The following topics discuss the special elements in Logi Info and Logi Report for use with this database:

- Connecting to HP Vertica
- Retrieving Data from HP Vertica

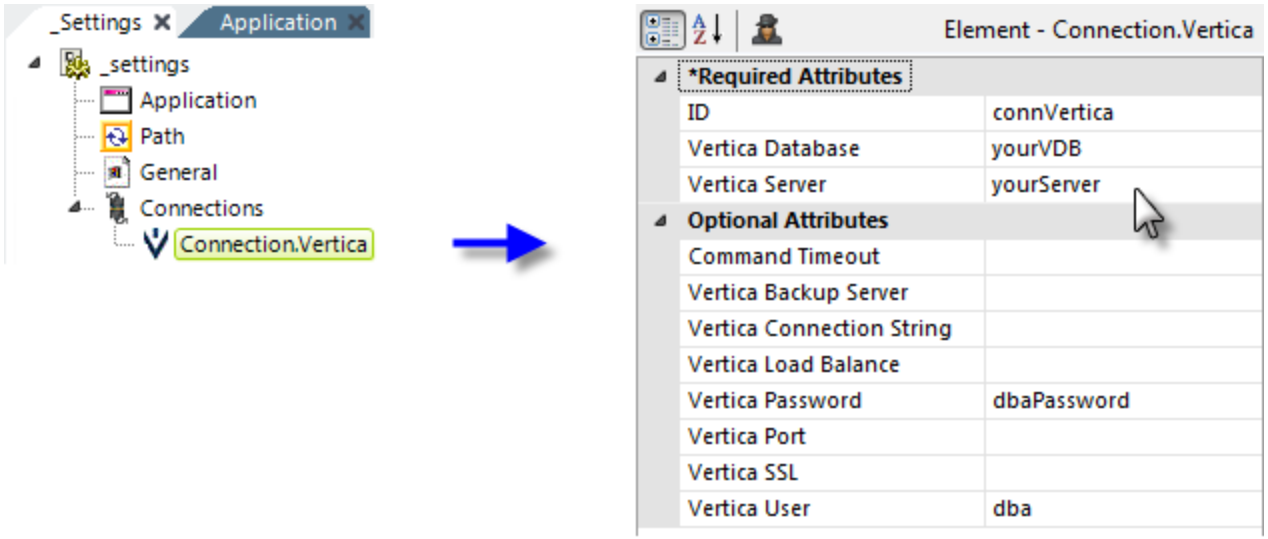
General information about HP Vertica can be found at the official website.

About HP Vertica

The column-oriented HP Vertica RDBMS is designed to manage large, fast-growing volumes of data and provide very fast query performance when used for data warehouses and other query-intensive applications. It claims to drastically improve query performance over traditional relational database systems, to provide high-availability, and offer petabyte-scalability on commodity enterprise servers. A free, feature-limited copy for local use is available for download and subscriptions are available to their hosted service.

Connecting to HP Vertica

In order to use HP Vertica, you need to use a specific Connection element, **Connection.Vertica**, to connect your Logi application to a Vertica database.



As shown above, the **Connection.Vertica** element is added to the **_Settings** definition, beneath the Connections element. Its attributes are then set appropriately for the Vertica datasource. The **Vertica Parameters** element, a child of the connection element, can be used to supply additional parameters for connecting to the Vertica database.

Attributes

The Connection.Vertica element attributes are:

Attribute	Description
ID	(Required) Specifies an element ID, unique across the application.
Vertica Database	(Required) Specifies the name of the Vertica database to be accessed.
Vertica Server	(Required) Specifies the name of the Vertica server to be accessed.
Command Timeout	Specifies the amount of time, in seconds, before the request to connect to the database is presumed to have failed. The default value is *60* seconds.
Vertica Backup Server	Specifies a string containing the host name or IP address of one or more alternate Vertica servers to connect to if the primary Vertica server connection fails. Multiple servers may be specified in a comma-separated list. The host name or IP address can also include a colon followed by the port number for the database. The default port is *5433*.
Vertica Connection String	Specifies a full JDBC connection string to the Vertica database. If specified, it will override all other attribute values for this Connection element. If any Vertica Parameter elements are present, they will be processed and added to the connection string.
Vertica Load Balance	Specifies whether the client will allow its connection to be redirected to another host in the Vertica database. This setting only has an effect if the server has also enabled load balancing. The default value is *False*.
Vertica Pass-	Specifies the password associated with the Vertica database user name.

Attribute	Description
word	
Vertica Port	Specifies the Vertica database port address. The default value is *5433*.
Vertica SSL	Specifies whether to use SSL for this connection. The default value is *False*.
Vertica User	Specifies the Vertica database user name.

Special Drivers Required: Vertica 8.1

Vertica driver files can be found in the "driver folder": `<yourLogiAppFolder>\rdTemplate\rdDriver\Vertica\v8.1.x`

1. To use Vertica with Logi Info Studio, copy `Vertica.Data.dll` from the driver folder to

   ```
   C:\Program Files\LogiXML IES Dev\LogiStudio\bin
   ```

   The path above is for the default Logi Info installation, and will vary if you installed Logi Info in a folder other than the default installation folder.
2. For a .NET-based Logi application - Copy the file `Vertica.Data.dll` from the driver folder to `<yourLogiAppFolder>\bin`.
   For a Java-based Logi application - Copy the file `vertica-jdbc-8.1.1-0.jar` from the driver folder to `<yourLogiAppFolder>\WEB-INF\lib`.

Vertica 8.1 drivers are backwards compatible to Vertica server version 7.1.

Special Drivers Required: Vertica 7.2.3

Vertica driver files can be found in the "driver folder": `<yourLogiAppFolder>\rdTemplate\rdDriver\Vertica`.

1. To use Vertica with Logi Info Studio, copy `Vertica.Data.dll` from the driver folder to

   ```
   C:\Program Files\LogiXML IES Dev\LogiStudio\bin
   ```

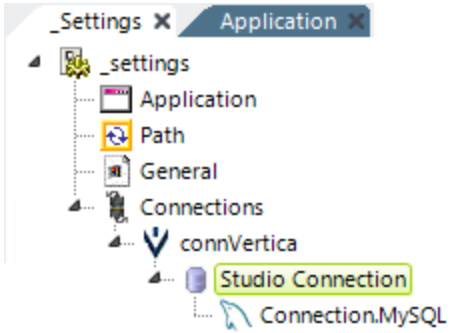
   The path above is for the default Logi Info installation, and will vary if you installed Logi Info in a folder other than the default installation folder.

2. For a .NET-based Logi application - Copy the file `Vertica.Data.dll` from the driver folder to *<yourLogiAppFolder>*`\bin`.
   For a Java-based Logi application - Copy the file `vertica-jdbc-7.2.3-0.jar` from the driver folder to *<yourLogiAppFolder>*`\WEB-INF\lib`.

Vertica 7.2 drivers are not backwards compatible with earlier versions.

Special Drivers Required: Vertica Versions Prior to 7.2.3

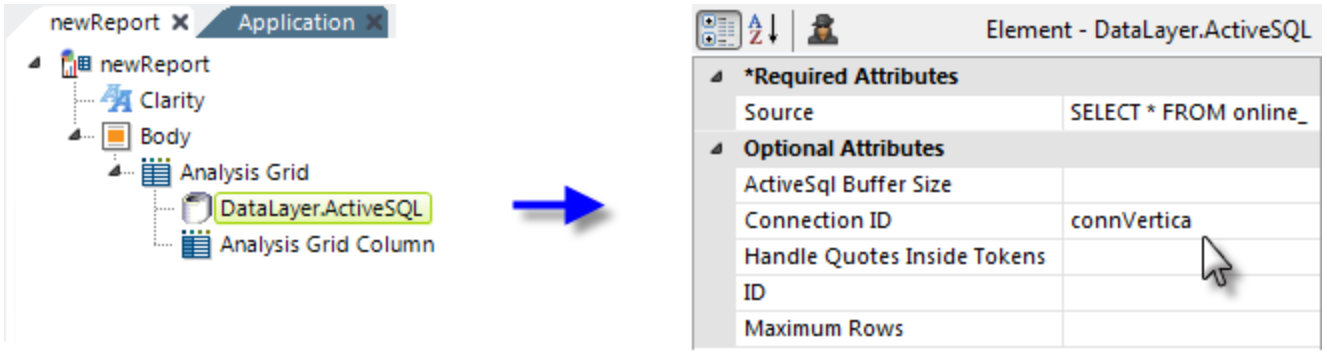
For a **.NET** Logi application, you will need to *download* the Vertica **ADO.NET driver** and place it in the `<yourLogiApp>\bin` folder. The driver is available from the My Vertica community web site, which requires you to register and login. Version 6.1.3.0 or later is required.
For a **Java** Logi application, you will also to *download* the Vertica **JDBC driver** and place it in the `<yourLogiApp>/WEB-INF/lib` folder. The driver is available from the My Vertica community web site, which requires you to register and login. Version 6.1.3.0 or later is required. In either case, if you want to be able to use Logi Studio's **Database Browser** and **SQL Query Builder** tools while working with Vertica, you must download the Vertica ADO.NET driver mentioned above and place it in the `bin` folder of your Logi Info installation, typically `C:\Program Files\LogiXML IES Dev\Logi Studio\bin`. If use of the Vertica ADO.NET driver is *not* possible, you can use a **Studio Connection** element to provide a separate connection for use by Studio's tools:



The Studio Connection element has no attributes and is simply a parent element for another non-Vertica Connection element, as shown above. That connection element is used by Studio's tools, such as the Database Browser, Web Metadata Builder, etc., during development instead of the primary connection that's used by the Logi Engine to access data at runtime.

Retrieving Data from HP Vertica

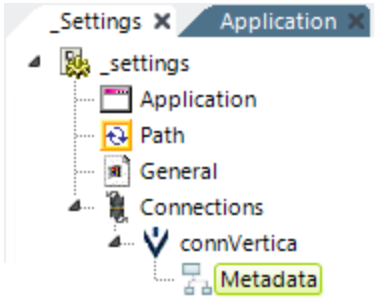
You can use DataLayer.SQL to retrieve data from Vertica but, for best performance with very large datasets, we recommend the use of *DataLayer.ActiveSQL*.



Vertica uses **HP Vertica SQL** which is based on standard SQL syntax but has a number of very important differences that you should be aware of as you create your queries. Please refer to the HP Vertica SQL Reference for more information. After the data is retrieved, you may condition, filter, and shape it just as you would any other datalayer data.

Using the Active Query Builder and Connection-Related Metadata

When using an Analysis Grid with the **Active Query Builder**, users can select the tables and Joins to be used for analyses at runtime. To facilitate this, developers need to create and specify the metadata files to be used. "Web Metadata Builder" on page 136 provides information about creating these files.



To specify the metadata files to be used with a Connection.Vertica element, add one or more child **Metadata** elements beneath it, as shown above. This element's attributes identify the metadata file and the name for it that will appear in the Active Query Builder interface.

Working with MongoDB

MongoDB is a cross-platform, document-oriented database system, classified as a "NoSQL" database.

The following topics discuss the special elements in Logi Info and Logi Report for use with this database:

- Connecting to MongoDB
- Retrieving Data from MongoDB
- Inserting Data into MongoDB
- Updating Data in MongoDB
- Removing Data from MongoDB
- Running MongoDB Commands

General information about MongoDB can be found at the official website.

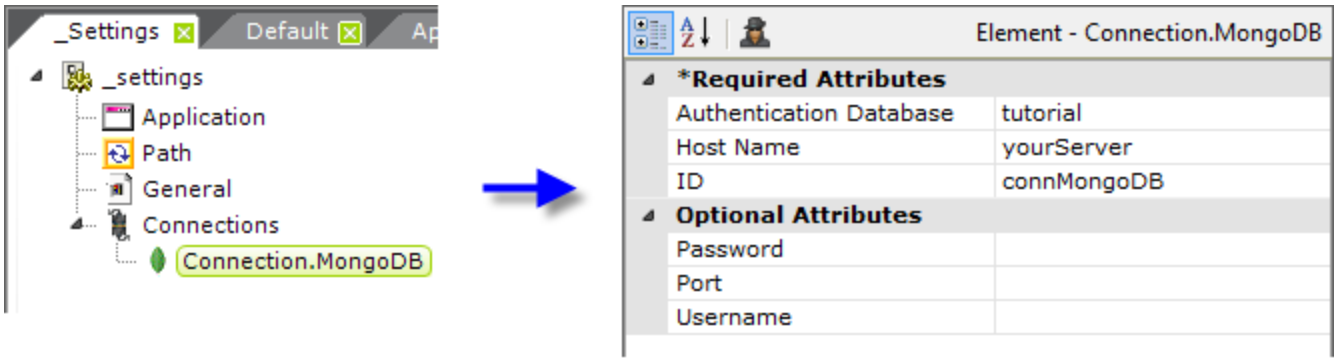
About MongoDB

MongoDB is a cross-platform, document-oriented database system, classified as a "NoSQL" database. Rather than using a traditional table-based, relational DB structure, it uses a collection of JSON-like documents with dynamic schemas and hierarchical data, called "BSON". The system is highly-scalable and lends itself to "big data" implementations.

Many Logi elements are designed to work with tabular data in rows and columns, so we've provided functionality to transform MongoDB hierarchical data directly into tabular XML data.

Connecting to MongoDB

The most recent .NET and Java drivers shipped with Logi Info v12.7 work with MongoDB 2.6 through 4.0. Logi Info v12.6 supports MongoDB 2.6 through 3.6. If you're using an older version of MongoDB, such as 2.4, you cannot upgrade your Logi application to Info v12.6. A specific Connection element, **Connection.MongoDB**, to used to connect to the database.



Note that the Authentication Database name is case-sensitive!

As shown above, the **Connection.MongoDB** element is added to the **_Settings** definition, beneath the Connections element. Attributes are set appropriately for the MongoDB datasource.

A **Mongo Params** element is available for use beneath Connection.MongoDB to supply additional options when connecting to the MongoDB database. Examples of additional parameters include "connectTimeoutMS" and "maxPoolSize". Refer to the MongoDB connection options documentation for a the complete list of options.

Retrieving Data from MongoDB

Special datalayer elements are used to retrieve data from the MongoDB system and their attributes can be used to form a "query" so that specific data is returned. In some cases, the data retrieved is limited to 16MB.

The available datalayers include the following (click the link for more information and examples):

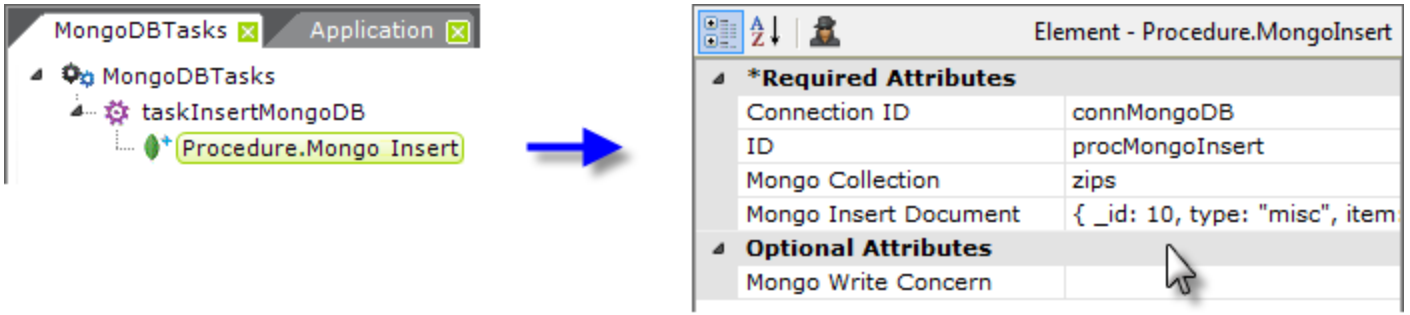
Element	Description
DataLayer.Mongo Find	Retrieves documents from a MongoDB collection using the MongoDB Find API.
DataLayer.Mongo Map Reduce	Runs a MongoDB map-reduce operation to return one or more documents.
DataLayer.Mongo Run Command	Runs a MongoDB command, suitable for use with the Aggregation Pipeline, a simpler alternative to using map-reduce operations, to return one or more documents.

MongoDB results are often hierarchical and may consist of multiple levels of parent-child relationships. In order to work with Logi elements, the data often must be "tabularized" and this can be done by using *The Flattener* element.

In many respects, these datalayers function exactly as other datalayer elements do and, once retrieved, the data can be filtered and conditioned using all the usual datalayer child elements.

Inserting Data into MongoDB

Inserting data into MongoDB is accomplished using a Process Task and the **Procedure.Mongo Insert** element.



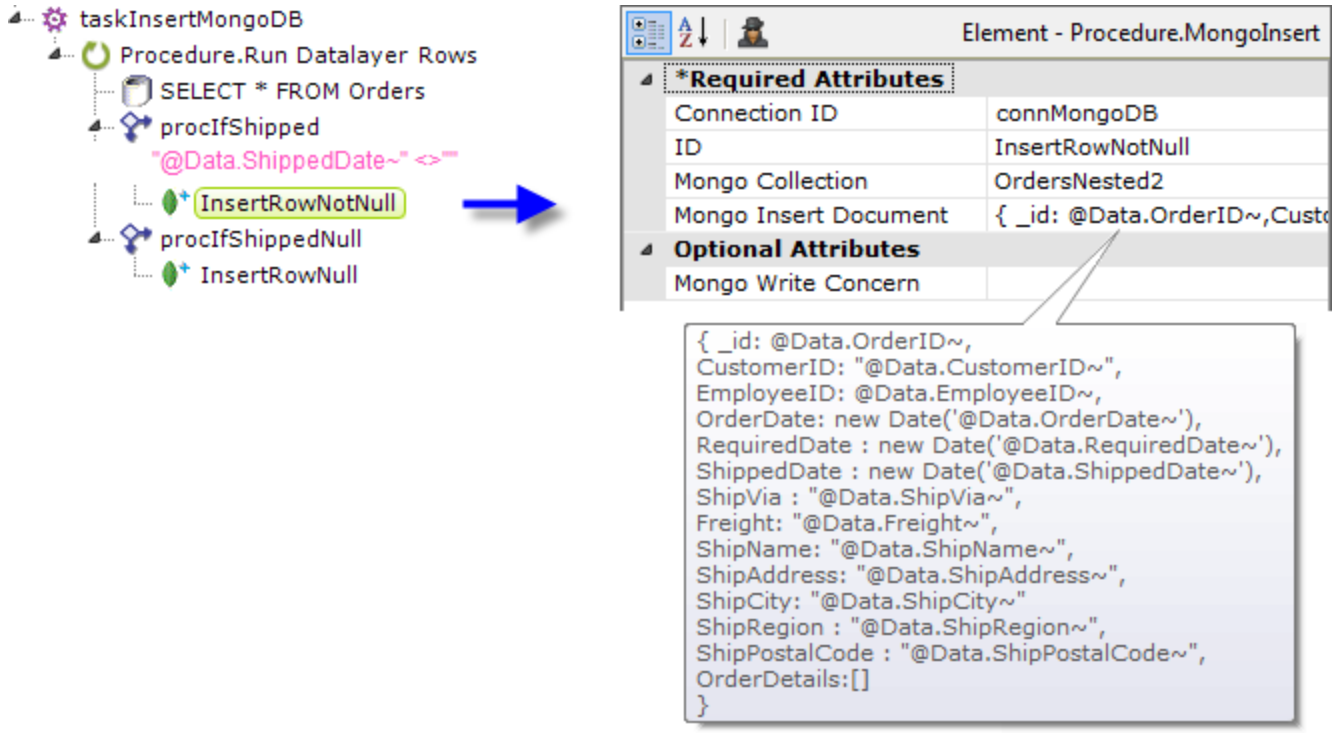
In the example above, a Process Task has been defined that includes a Procedure.Mongo Insert element and its attributes have been set to insert data into the "zips" collection.

The Procedure.Mongo Insert element's attributes include:

Attribute	Description
Connection ID	(Required) Specifies the ID of a Connection.MongoDB element in the application's _Settings definition.
ID	(Required) A unique element ID.
Mongo Collection	(Required) Specifies the name of the document collection to open. This value is *case-sensitive*.

Attribute	Description
Mongo Insert Document	(Required) Specifies the new document to be inserted to the collection. To vary the document content based on user or other input, use tokens, such as @Request, inside the document.
Mongo Write Concern	Specifies whether the procedure should wait for acknowledgement of the write operation. The default value is *Acknowledged*, meaning wait for acknowledgement.

No results are returned when the insert operation runs; however, an error that can be handled with the **If Error** element will be thrown if it fails. Let's examine a more complicated example:



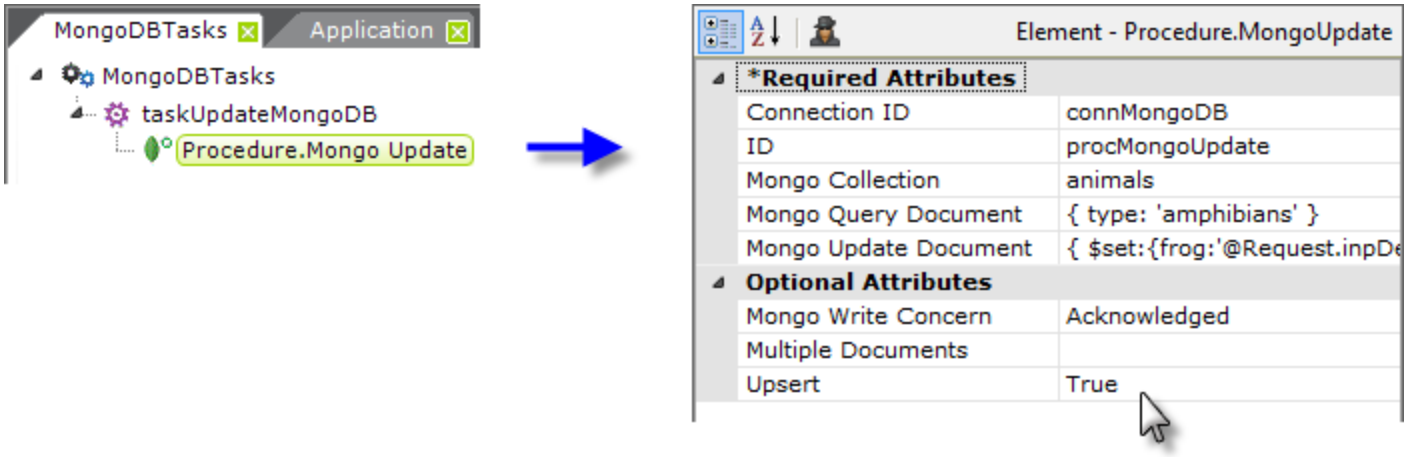
In the process task shown above, data is being read from a SQL database and inserted into a MongoDB collection. First, **Procedure.Run Datalayer Rows** and its datalayer query the SQL database. Then, for each datalayer row, Procedure.Mongo Insert is run. Two **Procedure.If** elements are used to determine if the SQL data's "ShippedDate" column is null or not, in order to run different insert elements with slightly different Mongo Insert Documents for each case.

Note how the @Data tokens from the SQL query data are used in the Mongo Insert Document.

The **Procedure.Mongo Update** element can perform an "Upsert", inserting a new document if one doesn't exist to be updated; see "Updating Data in MongoDB" on the next page.

Updating Data in MongoDB

Updating existing data in MongoDB is accomplished using a Process Task and the **Procedure.Mongo Update** element.



In the example above, a Process Task has been defined that includes a Procedure.Mongo Update element and its attributes have been set to update data into the "animals" collection. Note the use of a @Request token in the update document.

The Procedure.Mongo Update element's attributes include:

Attribute	Description
Connection ID	(Required) Specifies the ID of a Connection.MongoDB element in the application's _Settings definition.
ID	(Required) A unique element ID.

Attribute	Description
Mongo Collection	(Required) Specifies the name of the document collection to open. This value is *case-sensitive*.

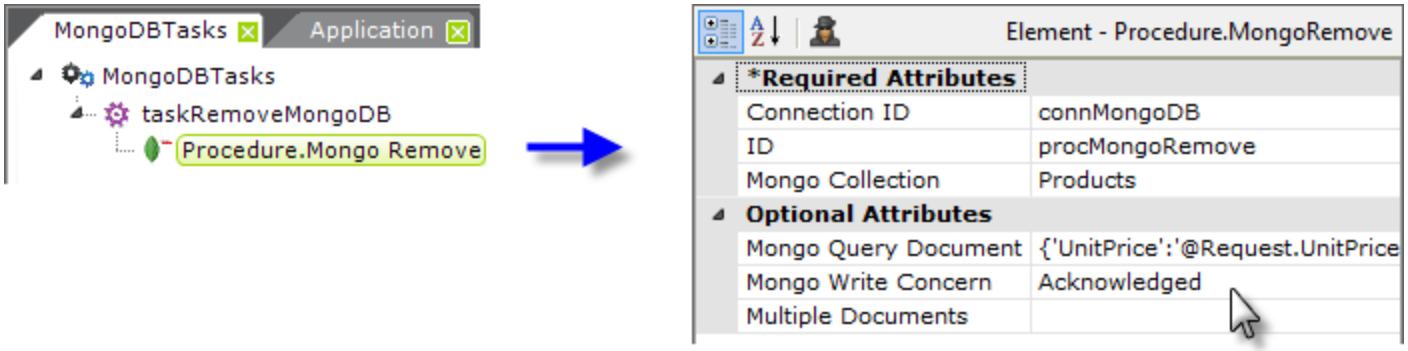
Mongo Query Document	Specifies a Mongo "find" document to identify one or more documents. Examples:
	Find all documents with "type" equal to "snacks".
	`{ type: "snacks" }`
	
	Find all documents with "type" equal to "food" or "snacks"
	`{ type: { $in: [ 'food', 'snacks' ] } }`
	To vary the document content based on user or other input, use tokens, such as @Request, inside the document.

Mongo Update Document	(Required) Specifies the new document to be updated, or possibly inserted into to the collection. To update selected attributes, instead of replacing the entire document, use the Mongo "$set" operator. To vary the document content based on user or other input, use tokens, such as @Request, inside the document.
Mongo Write Concern	Specifies whether the procedure should wait for acknowledgement of the write operation. The default value is *Acknowledged*, meaning wait for acknowledgement.
Multiple Documents	Specifies whether this command should affect multiple documents. The default value is *False*.
Upsert	Specifies whether this update command should be performed as an "Upsert". When the key already exists, the item is updated, otherwise a new item is inserted. The default value is *False*.

After an update procedure runs, you can use the special token @Procedure.*myProcMongoUpdateID*.DocumentsAffected~ to report the number of documents actually updated or inserted by an update operation.

Removing Data from MongoDB

Removing existing data from a MongoDB is accomplished using a Process Task and the **Procedure.Mongo Remove** element.



In the example above, a Process Task has been defined that includes a Procedure.Mongo Remove element and its attributes have been set to delete any documents in the "Products" collection that include a UnitPrice equal to the value of a @Request token.

The Procedure.Mongo Remove element's attributes include:

Attribute	Description
Connection ID	(Required) Specifies the ID of a Connection.MongoDB element in the application's _Settings definition.
ID	(Required) A unique element ID.
Mongo Collection	(Required) Specifies the name of the document collection to open. This value is *case-sensitive*.

Attribute	Description

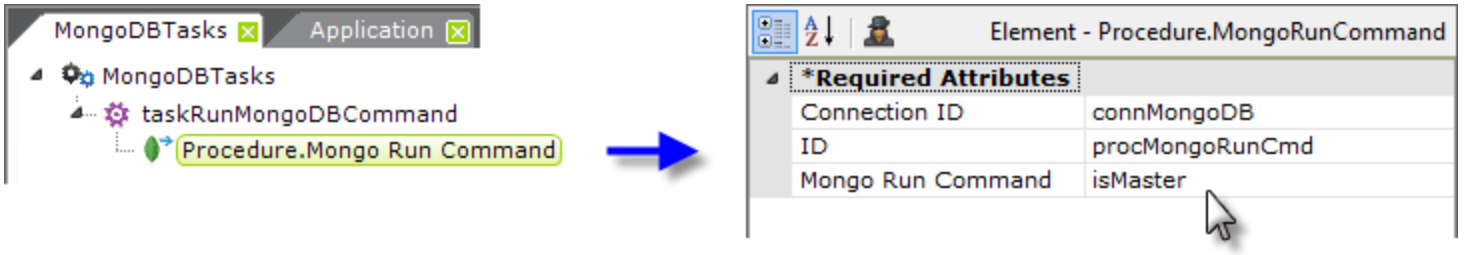
Mongo Query Document	Specifies a Mongo "find" document to identify one or more documents. Examples:
	Find all documents with "type" equal to "snacks".
	`{ type: "snacks" }`
	
	Find all documents with "type" equal to "food" or "snacks"
	`{ type: { $in: [ 'food', 'snacks' ] } }`
	To vary the document content based on user or other input, use tokens, such as @Request, inside the document.

Mongo Write Concern	Specifies whether the procedure should wait for acknowledgement of the write operation. The default value is *Acknowledged*, meaning wait for acknowledgement.
Multiple Documents	Specifies whether this command should affect multiple documents. The default value is *False*.

After a remove procedure runs, you can use the special token @Procedure.*myProcMongoRemoveID*.DocumentsAffected~ to report the number of documents actually removed by the operation.

Running MongoDB Commands

You can issue MongoDB Commands using a Process Task and the **Procedure.Mongo Run Command** element.



In the example above, a Process Task has been defined that includes a Procedure.Mongo Run Command element and its attributes have been set to run the MongoDB "isMaster" command.

The Procedure.Mongo Run Command element's attributes include:

Attribute	Description
Connection ID	(Required) Specifies the ID of a Connection.MongoDB element in the application's _Settings definition.
ID	(Required) A unique element ID.
Mongo Run Command	Specifies a MongoDB command. Refer to the MongoDB Database Command documentation for a the complete list of commands.

No results are returned when the command runs; however, an error that can be handled with the **If Error** element will be thrown if it fails.

Glossary

A

API

API, short for Application Program Interface, is a set of routines, protocols, and tools for building software applications. In business intelligence, APIs may be used to enable end-users to directly update source systems.

Authentication

Authentication is the verification of a user's identity.

Authorization

After a user's identity has been authenticated, authorization grants or denies access to reports, columns, and records to selected users or user-groups.

B

Big Data

Refers to both the ever-growing volumes of data in use today and also to services that are specifically engineered to provide and manipulate very large data volumes.

Business Analytics

Business analytics, or business intelligence (BI), gives customers the ability to rapidly create scalable, interactive data analysis applications, and self-service capabilities users can access from anywhere and on any device.

C

Columnar Data Store

Columnar data store is a type of big data repository containing structured data in columns and rows. The main benefits are that the data can be highly compressed and is easily searchable.

CRM

A Customer Relationship Management (CRM) system is a database-based system that records a company’s daily customer-related transactions. CRMs can help customer representatives to provide better service, close more deals, and increase revenue.

CSS

Cascading Style Sheets (CSS) is a technology that allows the presentation aspects of web pages to be separated from the page content. It can be used to add "styling" (e.g. apply fonts, colors, alignment, spacing, and more) to web pages.

D

Data Discovery

Data discovery is the capability to analyze data on-the-fly and uncover insights from it.

Data Enrichment

Data enrichment is a method of preparing data to make it ready for analysis and exploitation, and can include formatting, adding calculations, joining with other data, and more.

DevNet

The Logi Developer Network website.

Drill Down

Drill Down is a capability that allows the user to get a view of the underlying or supporting data used in an analysis.

Drill Through

Drill Through is similar to Drill Down but takes it one step further by applying analysis to the underlying or supporting data.

E

Elemental Development

A development approach used in Logi Info that lets developers build feature-rich applications by using reusable, pre-built elements, rather than by writing low-level code.

F

Forecasting

A technique involving data mining and analysis leading to predictions about what will happen in the future.

G

Geo Mapping

The combination of geographic and other data to produce map visualizations, such as Google or Leaflet maps.

H

Heatmap

A Heatmap chart, sometimes called a "tree map", which uses a unique arrangement of rectangles to represent data and relationships, using color and size.

I

Interpolation

The process of evaluating a literal value match containing one or more placeholders, yielding a result in which the placeholders are replaced with their corresponding values.

J

JavaScript

JavaScript is a programming language supported by the majority of modern web browsers and used by many websites.

JDBC

Java Database Connectivity (JDBC) is an API used to access relational databases. Open Database Connectivity (ODBC) is a similar API designed for use with Java.

JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format that's easy for humans to read and write, and easy for computers to parse and generate.

K

KPI

Key Performance Indicators (KPIs) are visual indicators, in the form of color-coded shapes, which are tied to a pre-defined, critical threshold.

L

LDAP

The Lightweight Directory Access Protocol (LDAP) is an Internet protocol applications use to look up information from a server and is frequently used for containing user login information.

M

My Term

My definition

N

NoSQL

"Not only SQL" (NoSQL) is an alternative to traditional relational databases, and doesn't rely on tables and a pre-determined schema. NoSQL databases are especially useful for working with large sets of distributed data.

O

ODBC

Open Database Connectivity (ODBC) is an API used to access relational databases. Java Database Connectivity (JDBC) is a similar API designed for use with Java.

OLAP

Online Analytical Processing (OLAP) is the process of analyzing data stored in multi-dimensional "cubes".

R

REST

Representational State Transfer (REST) is a type of API used to provide interoperability between computer systems on the Internet.

S

SSM

The Self-Service Module (SSM) is a package that includes Logi Info + SSRM + Discovery or Logi Platform Services.

SSRM

The Self-Service Reporting Module (SSRM) is a Logi Info add-on module that adds special elements to Info and includes the InfoGo application.

W

Write-Back

The ability to update data sources, typically by adding, editing, or deleting data.