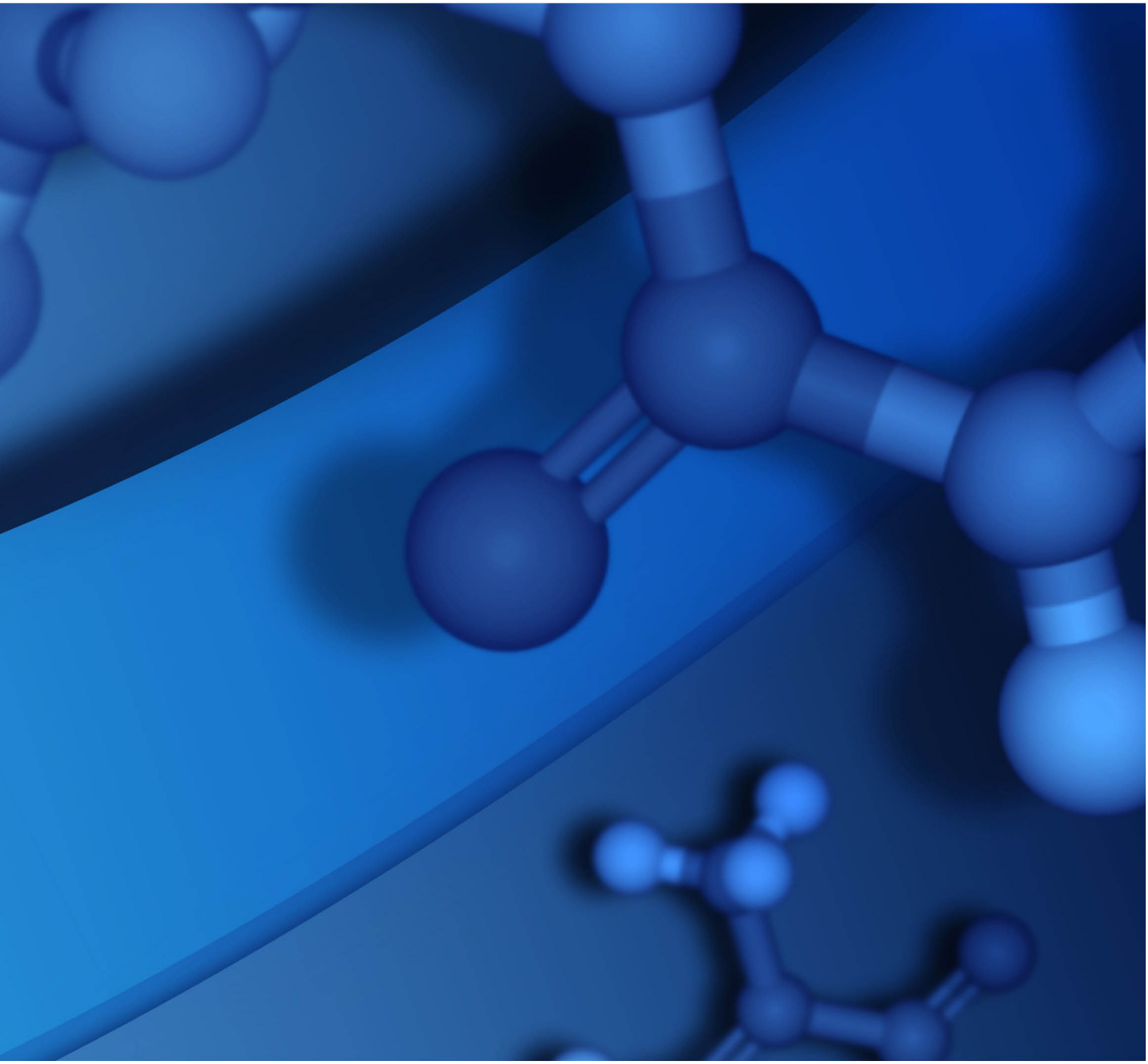
DASSAULT SYSTEMES

BIOVIA

ADMINISTRATION GUIDE

BIOVIA QUERY SERVICE 2019



Copyright Notice

Acknowledgments and References

To print photographs or files of computational results (figures and/or data) obtained by using Dassault Systèmes software, acknowledge the source in an appropriate format. For example:

> "Computational results were obtained by using Dassault Systèmes BIOVIA software programs. BIOVIA Query Service was used to perform the calculations and to generate the graphical results."

Dassault Systèmes may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to Dassault Systèmes Customer Support, either by visiting https://www.3ds.com/support/ and clicking **Call us** or **Submit a request**, or by writing to:

Dassault Systèmes Customer Support
10, Rue Marcel Dassault
78140 Vélizy-Villacoublay
FRANCE

Contents

Chapter 1:
Introduction

The BIOVIA Query Service is a Web service that provides client and Web applications with integrated access to corporate scientific data sources. It is supplied with BIOVIA Foundation. The Query Service is an HTTP-based service implemented using a REpresentational State Transfer (REST) interface. To learn more about the REST interface, see the *BIOVIA Query Service Developer's Guide*.

This guide is designed for BIOVIA Foundation and database administrators familiar with Oracle or SQL Server administration concepts, and BIOVIA Pipeline Pilot developers familiar with SQL and objects in a database.

Purpose of the BIOVIA Query Service

Traditional scientific database systems load data into individual schemas, each system having its own client application. As the availability of external data grows, or companies merge systems, the number of database systems and platforms multiplies, and more systems are introduced to handle the increase in data. The complexity of the infrastructure increases, and data becomes difficult to manage and access across multiple systems and platforms.

The Query Service solves the problem of searching across multiple data sources, by mapping the metadata of each data source to descriptive field aliases. Queries are built using these field aliases, and can be mapped onto any data source for which a corresponding mapping is provided. This method eliminates the need to migrate data or modify database schemas, and provides the flexibility to modify data sources.

Query Service queries are written using the Unified Query Language (UQL). UQL has a similar syntax to the widely-used Structured Query Language (SQL). An XML query format is also available. For a guide to UQL and to the XML query format, see the *BIOVIA Query Service Developer's Guide*.

The Query Service is compatible with Oracle and Microsoft SQL Server databases.

Chemistry searching in Oracle: Searching of chemistry data is possible with Oracle databases that have the BIOVIA Direct cartridge installed. Chemistry searching is not available for SQL Server databases. Chemistry data can, however, always be retrieved in results.

Query Service

Client Applications

The client application of the Query Service can be any of the following:

- Pipeline Pilot Client. Access to Query Service data sources is provided by a comprehensive set of Pipeline Pilot components.
- A web-based application that sends RESTful HTTP requests to the Query Service and converts the response data to HTML for display on web pages.
- A custom client with an HTTP back end, that presents the data retrieved from the Query Service to the user in any format desired.

The Query Service RESTful API

The Query Service exposes a REST API that developers of custom client and web applications can use to perform the following tasks:

- Get the metadata of a data source to determine what data can be queried
- Send queries in the body of HTTP POST commands, and retrieve references to lists of records
- Load and save lists of records
- Combine sets of results from queries, or search results
- Load and save data transformations and histories
- Execute cataloged (i.e. stored) queries

The client can be a web-based application or a custom client with an HTTP back end. The client requests Query Service resources via HTTP commands, and receives data in the body of the HTTP responses from the Query Service.

The Query Service can provide responses in various formats, including XML (the default format), XDfile, SDfile, JavaScript Object Notation (JSON), and several character-delimited formats. For more details about available reponse formats, see "Query result formats" in the *BIOVIA Query Service Developer's Guide*.

Key Features of the Query Service

With the Query Service, you can perform the following tasks:

- Search a diverse variety of data sources from a single web client or Web Port application regardless of underlying chemistry format, or database design.
- Easily integrate new and existing data sources for searching in BIOVIA Pipeline Pilot Server through the Query Service without modifying underlying database designs or migrating data.
- Build high-performance Web Port and Pipeline Pilot Server query and browse applications more easily and efficiently by using Query Service capabilities to simplify Pipeline Pilot protocol building and design.
- Use intrinsic list management capabilities in Pipeline Pilot Server to build Pipeline Pilot protocols to visualize, analyze, and report on search results from almost any data source.

Conventions

These are the conventions in this guide:

- References to functions, paths, and file names are presented in `this font`.
- For clarity and readability in paragraphs and tables, the font may also appear in bold.
- All code input and output snippets are presented `in this font` on a gray background.
- You must follow the order in a numbered list. A bulleted list has no order.

Note: In this guide, <`pps_install`> is used to represent the root folder of the BIOVIA Pipeline Pilot Server installation.
On 32-bit Windows this is typically `C:/Program Files/BIOVIA/PPS`
On 64-bit Windows this is typically `C:/Program Files/BIOVIA/PPS`
For 32-bit installations on 64-bit Windows this is typically `C:/Program Files (x86)/BIOVIA/PPS`

System Requirements

Query Service system requirements are covered in the *Pipeline Pilot Server 2019 System Requirements*.

Additional Information

For additional information about the Query Service, see the following documentation:

- *Query Service Developer's Guide*
- *Integrating Data Source (IDS) Guide*
- *Aggregating Data Source (ADS) Guide*

For more information about the Query Service and other BIOVIA software products, visit:
https://community.3dsbiovia.com

Chapter 2:
Overview for Administrators

This chapter provides an overview of the BIOVIA Query Service for administrators of the service.

Query Service Concepts

This section explains the main concepts behind the Query Service.

REpresentational State Transfer (REST)

REST is a model for distributed software design for client-server systems. It is characterized by the use of HTTP for communication between clients and servers, and by URLs that reflect the underlying structure of data and programming logic.

A software architecture that conforms to the principles of REST design is said to be RESTful. In a RESTful URL, objects and data elements on the server are typically represented as directories in a virtual file system, separated by slashes. Each directory is called a resource.

Data Sources

The Query Service relies on configuration files to define what data to access and how the data is to be organized.

End users see a list of data sources that they can use for querying, browsing, and reporting. These data sources are either **integrating data sources** (IDS) or **aggregating data sources** (ADS). Although these two types of data sources are different, they behave in the same way.

Administrators who must build these data source configuration files also use **relational data sources** (RDS). The relationships between these types of data sources are as follows: an IDS is always referencing an RDS. An ADS is referencing multiple IDSs which, in turn, reference their own RDSs.

- A relational data source (RDS) configuration defines the access to a schema in a relational database. An administrator defines the RDS in the Pipeline Pilot Admin Portal.

  Note: Relational data sources can also be defined in XML files at <pps_install>\web\queryservice\dsfiles. Your system may include old data sources that are defined in this way. However, **this method of defining a relational data source is deprecated**, and can present a security risk if a relational data source with the same name is configured in the Pipeline Pilot Admin Portal. For more information, see the *BIOVIA Query Service Administration Guide* > Required Configuration > Configuring Query Service data sources > "Adding and configuring relational data sources in the Pipeline Pilot Admin Portal".

- An integrating data source (IDS) configuration file describes how the relational data source tables are related and searched. It provides customized hierarchical access to a relational database. An administrator creates it using Data Source Builder.
- An aggregating data source (ADS) configuration file references two or more similar IDS files that can be searched together as a single data source. The various IDSs can be using the same or different database instances. An administrator creates it using Data Source Builder.

IDSs and ADSs can be queried by client applications using the Query Service. Client applications are BIOVIA Pipeline Pilot Client, BIOVIA Workbook, and others. For more information about setting up and configuring data sources, see the following:

- Configuring Query Service data sources
- *BIOVIA Query Service Developer's Guide*
- *BIOVIA Query Service Integrating Data Source Guide*
- *BIOVIA Query Service Aggregating Data Source Guide*

Data Source Builder

Data Source Builder is a Windows application that allows you to create IDS and ADS files in a graphical editor pane. You do not need to edit the XML of the IDS and ADS configuration files. Generally, using Data Source Builder is faster than editing data source XML files. Data Source Builder is included with BIOVIA Pipeline Pilot Server, and users can download it from the Pipeline Pilot Server home page. For instructions on installing it, see Installing Data Source Builder.

Query

A query is a command that searches and retrieves data according to particular search criteria. The queries used with the Query Service are written in Unified Query Language (UQL), which is a language that includes a standard set of expressions for constructing queries on different data sources. For information about UQL, see “Unified Query Language” in the *Query Service Developer’s Guide*. It is also possible to define a query in an XML format. This format is described in “Constructing an XML query” in the *Query Service Developer’s Guide*.

UQL queries are sent to REST resources as the payloads of HTTP POST methods. The server creates a query object and sends the client application the ID of that object as part of the response. An application can then send requests using that ID to retrieve the query results.

Active Query

An active query is an instance of a query that is stored temporarily on the server after the query has been executed. The active query includes a list of primary keys for the query results records. It has a unique identifier, so that it can be re-used in future requests.

Hitlist

A hitlist is the product of a successful query. An AQS hitlist contains the query definition + primary keys + selected fields + sort specification + the run date and time + metadata.

Note: There are different types of hitlists: AQS, Isentris XML External list, and Isis list. BIOVIA recommends using the AQS list. The others are from legacy applications.

Catalog

The Query Service catalog is a stored collection of definitions for queries, data transformations, and histories of operations on query results, that the Query Service can execute. Catalog data is stored in an XML file. The catalog REST resource provides information about the Query Service catalog.

Note: This catalog is not the same as the Pipeline Pilot Server Catalog which plays a role in indexing and searching in Pipeline Pilot Server for all components, protocols, and their parameters.

Metadata

Metadata is information about the contents of a data source, a query, or results. For data sources, the metadata includes the name and type of fields in a field hierarchy. The metadata determines the types of data that can be searched and browsed on a data source. For results, the metadata includes information about the fields included with the results, along with each field's display name, type, and other information.

Sharing Data Sources

A single data source can be used by many users. Typically, each user accesses a data source for a short amount of time. There is no blocking or waiting unless the total resources of the RDBMS have been exceeded.

Shared data sources are shared connections, allowing anyone to access the connection that is already opened and initialized. This means that, assuming one of the connections is available and sharable, you can start using it without the overhead of parsing the connection information and opening a data source.

If the data source is not sharable or if no available sharable connections are present, the data source is opened from scratch, taking a few seconds more in most circumstances and possibly longer for some complex or remote cases.

After the user is finished with an unshared data source, if it is not explicitly closed, it expires and gets closed in 15 minutes, releasing the resources. Shared data sources can also expire, but they do so after two hours.

The purpose of shared data sources is to reduce the amount of time it takes to perform queries on data sources that are commonly used and which have the same metadata for all users. BIOVIA recommends that you do not share data sources that are used infrequently. You should not to reserve resources for infrequent use or for data sources that have variable metadata. Sharing a data source with variable metadata would present invalid metadata in some scenarios.

Data sources with variable metadata are uncommon. Data sources with variable data are typically ones that use security measures or custom initializers to present a different view of the data to different users.

For information about setting properties that enables data source sharing, see Setting Properties for Shared Data Sources.

Closed Data Sources

The Query Service uses an algorithm in a resource monitor to determine when a data source should be closed. The algorithm includes the amount and type of usage of the data source in the computation. The resource monitor runs in the background and monitors data sources and recordsets. When the resource monitor determines that a data source has been used "enough", for example, a certain number of recordsets have been created, the data source is marked as "not to be used" and a fresh copy of the data source is opened. These marked copies of the data source stay open until all their dependent resources are closed, and then they are closed.

Note: A recordset opened with one copy of a data source can be used later with another copy of the data source.

The Lifetime of a Recordset

When a query's search is complete, the QueryID property contains an identifier for the query. The QueryID can be used to retrieve information about the query or data from the query. The data includes the query's recordset.

The query is valid for the lifetime of a server. If the server is restarted, you start fresh.

The life of the recordset defaults to five minutes (unshared data sources) or 2 hours (shared data sources). Send a reference — any reference — to reactivate the recordset.

A recordset for a shared data source is also refreshed after it has been accessed 400 times.

To remove a recordset permanently, you must use the REST endpoint query/{query ID}.

To remove a recordset permanently:

- Run the Query Service component *Query Delete*.
- Use the `delete` HTTP method of the query's REST endpoint query/{query ID}.

Managing Lists and Temporary Tables

The Query Service creates temporary tables for list handling depending on the operation and the parameters of the operation.

The temporary tables are created in the following locations:

- **Oracle:** By default, the temporary tables are created in the schema that is used to access the data. You can override this behavior by specifying a user name and password to use for creating internal lists as Oracle tables. Reasons for doing so:
  - You want to manage these temporary tables at a central location for efficient list handling and cleanup.
  - You do not want to grant CREATE TABLE privileges to the schemas used to access the data.

  To create internal lists as Oracle database tables, specify **List DB Username** and **List DB Password** in the **Query Service Settings** when using the **Add Data Source** or **Edit Data Source** form in the **Setup > Data Sources** section of the Pipeline Pilot Admin Portal. If you leave these blank, temporary tables are created using the schema defined by the **Optional DB Username** in the same form.

  For more information about these settings, see Configuring Query Service Data Sources.
- **SQL Server:** Temporary tables are always managed in the tempdb database.

Configuration

Additional ways to control Query Service are as follows:

You can change the memory for the Apache Tomcat which affects the Query Service.

- From the Pipeline Pilot Admin Portal, go to **Maintenance** > **Manage Server**. You can change the maximum and minimum settings. The default maximum is Xmx512m, which means 512 megabytes. The default minimum is Xms128m, which means 128 megabyte. If you change either one, you must

click **Restart Tomcat** before the changes take effect.

```
-Xms128m
-Xmx512m
-Xss512k
-XX:MaxPermSize=128m
-Dorg.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false
-Dfile.encoding=UTF8
```

You can change the following settings for the Query Service:

- Logging level
- Timeout

For more information about these settings, see Configuring Query Service in the Administration Portal.

Important Files and Folders

As the administrator, you need to know about these files that are required for the operation of the Query Service:

- templates – These are data sources for the BIOVIA products like databases, or applications. They are located in the following location:
  Windows:
  `<pps_install>\web\queryservice\isentris\system\config\datasource\templates`
  Linux:
  `<pps_install>/web/queryservice/isentris/system/config/datasource/templates`
- `dsfiles` folder – For BIOVIA Pipeline Pilot Server installations, this is the folder where all of the active data source configuration files are stored.
  Windows:
  `<pps_install>\web\queryservice\dsfiles`
  Linux:
  `<pps_install>/web/queryservice/dsfiles`
- `dsfiles` folder in BIOVIA Isentris installation – You might want to copy some data source configuration files from the Isentris server to BIOVIA Pipeline Pilot Server. The `dsfiles` folder for Isentris is in this location:
  Windows:
  `{Isentris installation directory}\system\config\datasource\dsfiles`
  Linux:
  `{Isentris installation directory}/system/config/datasource/dsfiles`
- `Catalog.xml` file – The `Catalog.xml` file defines the Query Service catalog. The catalog is a repository of predefined query and data transformation definitions. The `Catalog.xml` file is deployed here:
  Windows:
  `<pps_install>\apps\scitegic\queryservice`
  Linux:
  `<pps_install>/apps/scitegic/queryservice`
- Query Service log – The query service log is specific to Query Service; only the Query Service writes to it.
  Windows:
  `<pps_install>\logs\messages\javaserver_queryService.log`

Linux:
<pps_install>/logs/messages/javaserver_queryService.log

Chapter 3:
Required Configuration

This chapter provides instructions on setting up the Query Service. The Query Service is installed with BIOVIA Pipeline Pilot Server. Within Pipeline Pilot Server, the Query Service is managed by the Apache Tomcat servlet container.

For optional configuration tasks, see Optional Configuration Tasks.

The following tasks are required for Query Service to work:

- Configuring Query Service Permissions
- Configuring Data Sources to be accessed by the Query Service. This step is mandatory for all installations of the Query Service.
- Configuring Query Service in the Administration Portal
- Configuring the Query Service catalog that contains predefined query definitions that are used with the Query Service. This step is only necessary if your users are using the predefined query templates, transforms, and histories in the catalog.

Host Name and Port Name

Every Query Service Universal Resource Identifier (URI) begins with the host name and a port number of the Query Service. You configure the port numbers when you install Pipeline Pilot Server. Two ports are configured:

- An HTTP port. The default HTTP port number is 9944.
- An HTTPS port. The default HTTPS port number is 9943. Encryption within HTTPS is intended to provide benefits like confidentiality, integrity, and identity.

Use these host names and port numbers in URIs that access Query Service resources.

Configuring Query Service Permissions

This section explains how to set or change user and administrator permissions for the Query Service.

You configure Query Service permissions in Pipeline Pilot Server or Foundation Hub. If Foundation Hub is connected to Pipeline Pilot Server, you can only configure permissions in Foundation Hub.

The following Query Service permissions are created when you install BIOVIA Foundation:

Pipeline Pilot Server permission	Foundation Hub permission	Default group with permission	Role
QueryService|Logon	QueryService/Logon	QueryService/Users	Has access to the non-administrative Query Service REST resources.
QueryService|Administer	QueryService/Administer	QueryService/Administrators	Has access to all Query Service REST resources, including the monitor/... and admin/... resources.

The default group membership of the Query Service groups is as follows:

Query Service group	Member groups
QueryService/Users	Platform/Everyone
QueryService/Administrators	Platform/Administrators

Therefore, by default, all Pipeline Pilot Server users are Query Service users, and all Pipeline Pilot Server administrators are Query Service administrators.

Notes:

- If Pipeline Pilot Server is using Foundation Hub for authentication, members of the Foundation/Administrators and Foundation/Users groups do NOT have access to the Query Service by default. To give them access, you must make these groups members of the QueryService/Administrators and QueryService/Users groups in the Foundation Hub **Admin and Settings** pages.
- If you are using Foundation Hub for authentication, you can only assign permissions to groups, not to individual users.

Configuring Permissions in Pipeline Pilot Server

To set or change user and administrator permissions for the Query Service in Pipeline Pilot Server:

1. In a browser, navigate to the Pipeline Pilot Server home page.
2. Click **Administration Portal**.
3. If prompted, enter the user name and password of a Pipeline Pilot Server administrator.
4. If appropriate, create new users for the Query Service in the **Security > Users** page.
5. In the **Security > Groups** page or the **Security > Users** page, assign users to the QueryService/Users or QueryService/Administrators groups.

   You can optionally create new groups with custom permissions.
6. In the **Security > Groups**, **Security > Permissions**, and **Security > Users** pages, assign the QueryService|Logon and QueryService|Administer permissions to groups and users, as appropriate. Instructions are provided on these pages, and in the Pipeline Pilot Admin Portal Help.

Note: If Pipeline Pilot Server is connected to Foundation Hub, the **Groups**, **Permissions**, and **Users** security pages are not accessible. In this case, you must either configure groups, permissions, and users in Foundation Hub, or disconnect Foundation Hub from Pipeline Pilot Server and then configure groups, permissions, and users in Pipeline Pilot Server.

Configuring Permissions in Foundation Hub

To set or change user and administrator permissions for the Query Service in Foundation Hub:

1. In a browser, navigate to the Foundation Hub home page.
2. If prompted, enter the user name and password of a BIOVIA Foundation administrator.
3. Click **Admin and Settings**.
4. If appropriate, create new users for the Query Service in the **Security > Users** page.
5. In the **Security > Groups** page, assign users to the QueryService/Users or QueryService/Administrators groups.

   You can optionally create new groups with custom permissions.

6. In the **Security > Groups** and **Security > Permissions** pages, assign the QueryService/Logon and QueryService/Administer permissions to groups, as appropriate. For instructions, click the **Application Help** icon.

   Note: In Foundation Hub, you can only assign permissions to groups, not to individual users.

Configuring Query Service Data Sources

Each Query Service data source must be configured before it can be used:

- You configure relational data sources in the Pipeline Pilot Admin Portal.
- You create integrating data sources in Data Source Builder. (You can also create IDSs in a text editor, but Data Source Builder offers many features that facilitate creation of IDSs, plus its own internal XML editor.)
- If you have existing Isentris data sources on an Isentris installation, you can use them with the Query Service. For information about how to use existing Isentris data source files, see Setting Up a Data Source Using Existing Isentris Data Source Files.

Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal

This section describes how to configure and manage relational data sources for use with the Query Service. A relational data source (RDS) configuration defines the access to a schema in an Oracle or SQL Server relational database. An administrator defines the RDS in the Pipeline Pilot Admin Portal.

IMPORTANT! Ensure that you have access to the data source before proceeding.

To configure a relational data source:

1. Open the Pipeline Pilot Server home page in a browser.
2. In the **Administrators** menu, click **Administration Portal**.
3. Enter the administrator user name and password and click **OK**. The Pipeline Pilot Admin Portal opens.
4. In the Pipeline Pilot Admin Portal, click **Setup > Data Sources**.
5. Click **Add Data Source** below the **Data Source List** table. (The list may be empty when you first view it.) The **Add Data Source** form opens to the right of the **Data Source** list.
6. Enter details of the data source in the **Add Data Source** form:

Field	Description
Name	Enter a name for the data source. The naming convention is as follows: `{data source name}_REL` For example: MyCompoundDb_REL

Field	Description
	IMPORTANT! Ensure that the name is not already being used by an XML relational data source published in the folder `<pps_install>\web\queryservice\dsfiles` Such an XML relational data source file might have been created for use with BIOVIA Isentris or with early versions of the BIOVIA Query Service. If a published XML relational data source has the same name as a Pipeline Pilot Server relational data source, its presence overrides any access restrictions on the Pipeline Pilot Server data source. This situation creates a potential security loophole, and should be avoided. For example: ■ Choose a different name for the new relational data source. For example, if the `dsfiles` folder contains an old Isentris relational data source file called `ACD_REL.xml`, do not name the new data source `ACD_REL`, but use an alternative such as `ACD_PPS_REL`. ■ Verify whether the XML relational data source is still needed, and if it is not, delete it. Note that XML RDS files are deprecated. You should create new relational data sources in Pipeline Pilot Server.
Description	Enter a textual description of the data source.
Type	From the list, select the data source type. You must select **JDBC**. This is the only data source type supported by the Query Service. Note that when you select **JDBC**, the layout of the form changes.
Access privileges	Access privileges for the data source. Click the + icon, and edit the form as appropriate. If you do not edit access privileges when you initially configure the data source, all Query Service users will have access to the data source. You may find it convenient to defer configuration of access privileges until you have verified that the data source is working. After you have verified that the data source is working, you can restrict access. You can restrict access to specific users or groups, and define levels of access for each of these. Access levels include **Edit Data Source**, **View Data Source**, **Use Data Source**, and **Deny Access**.
Driver	The database driver for the data source. ■ For Oracle databases, select **Oracle.jdbc.OracleDriver**. ■ For SQL Server databases, select **com.microsoft.sqlserver.jdbc.SQLServerDriver**.
Connection String	The database connection string. Use a string with the appropriate format for your database: **Oracle database using service name:** `jdbc:oracle:thin:@<server>:<port e.g. 1521>/<service_name>` **Oracle database using service ID:**

Field	Description

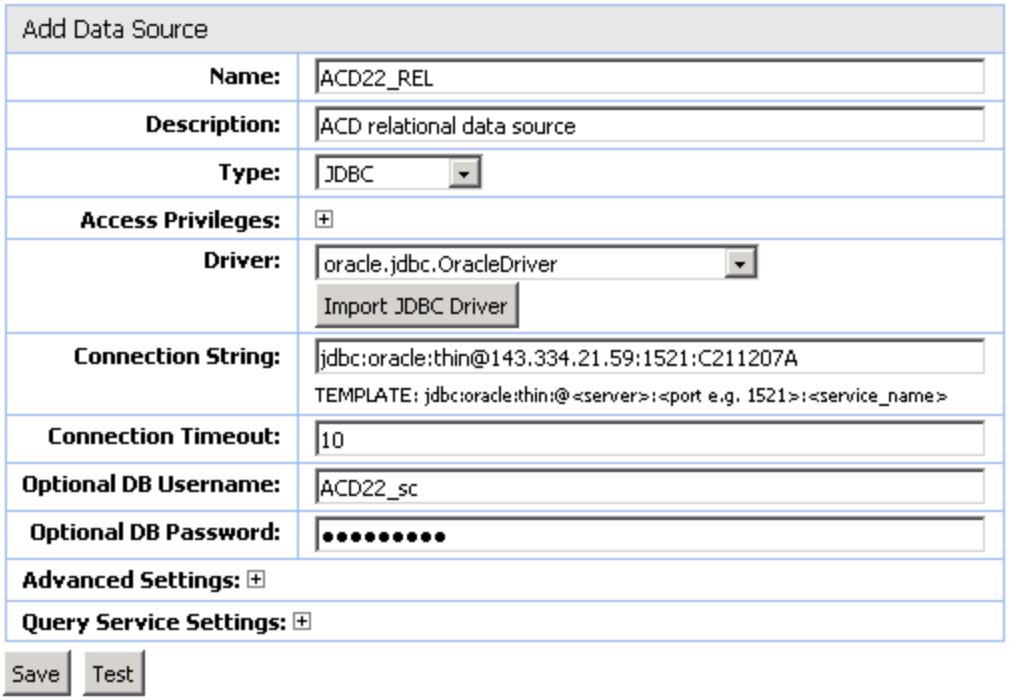
	`jdbc:oracle:thin:@<server>:<port e.g. 1521>:<service_id>`
	SQL Server database:
	`jdbc:sqlserver://<server_name>:<port e.g. 1433>;databaseName=<database_name>`
	The correct format of the string is indicated by the template below the text box. You may find it helpful to copy and paste the string, inserting appropriate values to replace the values in angled brackets. Your database administrator can provide you with the correct details.
	If you use the text in the template, make sure you remove all angled brackets.
	Examples:
	`jdbc:Oracle:thin:@MyServer:1521/MyServiceName` (Oracle with service name)
	`jdbc:Oracle:thin:@MyServer:1521:MySID` (Oracle with SID)
	`jdbc:sqlserver://myServer:1433;databaseName=myDatabase` (SQL Server)

Connection Timeout	This enables you to use JDBC/ODBC connection pooling. Specify a non-zero number of seconds for the connection to stay open. Any open connections are closed after this amount of time has elapsed since the last job finished in the pooled server.
Optional DB Username	THIS IS REQUIRED. The user name for the Oracle or SQL Server database. Consult your database administrator.
Optional DB Password	THIS IS REQUIRED. The password for the Oracle or SQL Server database.

Advanced Settings (Optional)	
Require PP Credentials	Not relevant for Query Service.
Proxy Authentication	Add a checkmark if you want to use Proxy Authentication for Oracle. A specific account is used to connect to the database and, separately, the current user's credentials are passed through for auditing. For more information, see "Using Oracle Proxy Accounts." This setting is not applicable to SQL Server data sources.
Initial SQL	Specify SQL statements that need to be executed whenever a connection is opened. You can include procedures (for example, PL/SQL with Oracle). One use for this field is to propagate the user name to the database for row-level security and audit purposes. To refer to the user, use the special string `$(username)`. For example: `{ call security.myProcedure('$(username)') }` **Note:** To specify multiple statements, use the appropriate database-specific conventions. For example, use the conventions `BEGIN ... END` with Oracle.
Required Registrant	Specify to limit access to a particular RDS. For example, if you specify scitegic/queryservice, the RDS can only be accessed by Query Service components. For more information, see "To limit usage of a data source by component" in the "Setup and Configuration" chapter of the *Pipeline Pilot Server Admin Portal Guide* in the **Administrators** tab of the Pipeline Pilot Help Center.
Application Info	Not relevant for Query Service.

Query Service Settings (Optional)	
List DB Username	**Oracle:** Specify the Oracle user name which is used by Query Service to create temporary tables for list handling. The user must have CREATE TABLE system privileges in Oracle. If you leave this box empty, temporary tables are created using the connection user name. **SQL Server:** Leave empty. SQL Server uses the tempdb database for temporary tables.
List DB Password	**Oracle:** Specify the password for the List DB Username. **SQL Server:** Leave empty.
Query Service Properties	Specify additional relational data source properties that you want for the Query Service, using the format `property name=property values` You can enter multiple properties using a semi-colon ; as a delimiter. For example, set `ALLOWBIGDECIMAL` to `True` if your database contains large integer values. A BigDecimal consists of an arbitrary precision integer with unscaled value and a 32-bit integer scale. If zero or positive, the scale is the number of digits to the right of the decimal point. If negative, the unscaled value of the number is multiplied by ten to the power of the negation of the scale. The value of the number represented by the BigDecimal is therefore (unscaledValue × 10-scale).

Example values are shown below.



7. Click **Test**. A login dialog box opens. If the connection is valid, the message "Login Successful" displays and the **Save** button becomes active.
8. Click **Save**. The data source is saved with its new settings, and is added to the **Data Source List**. The next step is to create a corresponding IDS in Data Source Builder. (You cannot create an IDS in the Pipeline Pilot Admin Portal.)

   You can also create aggregating data source (ADS) configuration files in Data Source Builder.

Using the Template_rel.xml File

Relational data sources that are configured in the Pipeline Pilot Admin Portal use the following template file to provide their default settings:

```
<pps_install>\web\queryservice\dsfiles\Template_rel.xml
```

You can edit this file in a text or XML editor, to change the default settings in all new Pipeline Pilot Server relational data sources.

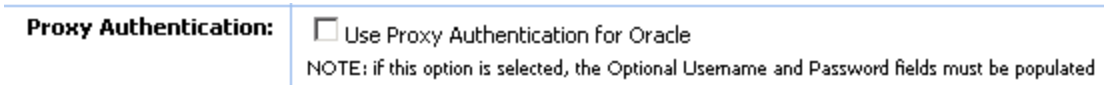
For details of the structure of the `Template_rel.xml` file, see "Relational Data Source Configuration Files" in the *BIOVIA Query Service Data Source XML Reference Guide*.

IMPORTANT! Take care when editing `Template_rel.xml`, as it affects all new relational data sources created in Pipeline Pilot Server.

Using Oracle Proxy Accounts

If you want more granularity in the use of credentials for logging into Oracle and being able to track individual users' Oracle transactions, you must configure Pipeline Pilot Server so that it uses Oracle proxy accounts. If you do not do this, multiple users of each relational data source will use the **Optional DB Username** credentials specified for that data source.

1. The first step to set up proxy accounts is for your database administrator to configure Oracle so that it allows Pipeline Pilot users to use Oracle proxy accounts. For more details, see the Oracle documentation.
2. Next, when you add a relational data source from the Pipeline Pilot Admin Portal — or when you edit a relational data source — select **Use Proxy Authentication for Oracle** in **Proxy Authentication**:



Note: Use of proxy accounts is not available for SQL Server data sources.

Editing a Relational Data Source

To edit the details of a relational data source in the Pipeline Pilot Admin Portal:

1. In the **Data Source List**, click the data source whose details you want to edit.
2. The **Edit Data Source** form opens. This form has the same appearance as the **Add Data Source** form, described in Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal.
3. Edit details of the data source, as appropriate.
4. Click **Test**. A login dialog box opens. If the connection is valid, the message "Login Successful" displays and the **Save** button becomes active.
5. Click **Save**.

The data source is saved with its new settings.

Deleting a Relational Data Source

To delete a data source in the Pipeline Pilot Admin Portal:

1. In the **Data Source List**, click the data source that you want to delete.

   The data source is highlighted in green, and a red cross appears to the right of it.
2. Click the red cross next to the data source. The data source is deleted.

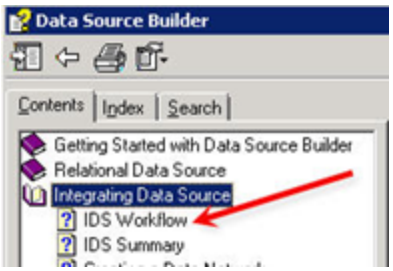
Installing Data Source Builder

Before you create an IDS, you must install Data Source Builder from the Pipeline Pilot Server home page. If you have not installed Data Source Builder yet, perform these steps:

1. Open the Pipeline Pilot Server home page in a browser.
2. In the **Administrators** menu, select **Install/Upgrade Data Source Builder**.
3. Read the instructions on the Data Source Builder installation page, and then click **Client Installer**. The Data Source Builder installer starts.
4. Follow the instructions in the Data Source Builder installer.

For instructions on using Data Source Builder, open Data Source Builder and click the help button. You can access topics from the **Contents** and **Index** tabs, or perform a search of terms in the **Search** tab.

For details about the IDS workflow in the Data Source Builder help, see the topic **Integrating Data Source > IDS Workflow**.



Creating an IDS

This section provides a summary of steps describing how to create IDS configuration files for use with the Query Service. IDS configuration files map relational database tables to hierarchical data nodes, so that relational databases can be queried in UQL or query XML statements.

Each Query Service IDS is mapped to a relational data source. Therefore, you must configure a relational data source before you create an IDS for it. If you have not already done this, follow the instructions in Adding and configuring relational data sources in the Admin Portal.

Because you control user access to the relational data source, BIOVIA recommends that you add and configure a different relational data source for each IDS that you build. You use Data Source Builder to create an IDS. For detailed instructions on using Data Source Builder, see the Data Source Builder Help.

Before you create an IDS, you should be familiar with the following:

- The tables and fields available in your relational data source.
- How IDSs work. For more information, please refer to the *Integrating Data Source Guide*, particularly the earlier chapters. That document explains the purpose of IDSs, and how they map relational database tables to hierarchical data nodes.
- (**Optional**) The syntax of UQL , which is the language for querying IDSs and ADSs. UQL is similar to SQL, but it represents all tables in a single hierarchy rather than through joins. Being familiar with the syntax of UQL is optional, but understanding it is useful.

IDS Workflow Summary

This section summarizes the main steps involved in creating and configuring an IDS in Data Source Builder.

This section does not go into detail about how to create a fully functional IDS. For full instructions on using Data Source Builder, see the Data Source Builder Help. For a guide to the structure of an IDS, see the *Integrating Data Source Guide*.

A good way of becoming familiar with IDSs is to study the templates for IDSs that are provided with Pipeline Pilot. These are provided for some of the databases most commonly used with the Query Service. Navigate to this location for templates:

Windows:

```
<pps_install>\web\queryservice\isentris\system\config\datasource\templates
```

Linux:

```
<pps_install>/web/queryservice/isentris/system/config/datasource/templates
```

Summary of Steps

1. Create relational data source definitions.

   Note: Only Pipeline Pilot Server administrators can create a new relational data source, from the page **Pipeline Pilot Admin Portal > Setup > Data Sources**.

2. Build an IDS.
3. Optimize the IDS.
4. Test the IDS.
5. Validate the IDS.

   Note: Any valid IDS can be opened even if the underlying relational data source is invalid or does not exist. The absence of a relational data source only limits editing.

6. Publish the IDS to the server.

Creating an ADS

You create aggregating data sources (ADS) in Data Source Builder. For instructions on creating an ADS, see the Data Source Builder Help. For a complete guide to ADSs, see the *Aggregating Data Source Guide*.

Using the Template IDS XML Configuration Files

BIOVIA Pipeline Pilot Server provides you with template XML configuration files for IDSs. These templates are for databases that are commonly used with the Query Service. You can study the template IDS configuration files in Data Source Builder as examples of how IDSs are constructed.

You can use the template IDS XML files as the basis of your own published IDSs, but you may have to edit them extensively so that they operate as desired with your data.

Note: In addition to template IDS files, some template relational data source XML files are provided with Pipeline Pilot Server. BIOVIA recommends that you do **not** use these as the basis of your own relational data sources, because they are not integrated into the BIOVIA Foundation security model. Instead, you should create relational data sources in the Pipeline Pilot Admin Portal, by following the instructions in Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal.

To set up an IDS from a template XML file:

1. Ensure that a relational data source is configured for the IDS template that you want to use. Configuration of relational data sources is explained in Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal.
2. **In Windows**, copy the template XML file for your database type from this location:

   ```
   <pps_
   install>\web\queryservice\isentris\system\config\datasource\templates\ids
   ```

   In Linux, copy the template XML file for your database type from this location:

   ```
   <pps_
   install>/web/queryservice/isentris/system/config/datasource/templates/ids
   ```

   to this location:

   ```
   <pps_install>/web/queryservice/dsfiles
   ```

   For example, if you want to set up an IDS for the ACD database, copy `ACD_IDS.XML`.

3. Open Data Source Builder. If Data Source Builder is already open, click the **Reload Data Sources** tool.

   The copy of the template IDS should be shown in the **Published Data Sources** list in Data Source Builder. If it is not shown, there is probably no relational data source configured for it. In this case, open the IDS XML file in a text editor, and check that the `sourceName` attribute of the `ConnectionAttributes` element is the same as the name of the relational data source in the **Name** column of the **Published Data Sources** list:

```
<DataSourcesByName>
  <DataSource name="ACD_IDS" type="IDS">
    <Connection>
      <ConnectionAttributes sourceName="ACD_REL" />
```

Setting Up a Data Source Using Existing Isentris Data Source Files

If you have data source files that have already been configured in an existing Isentris installation, you can make copies of those Isentris data sources to use with the Query Service. To set up a data source using existing Isentris data source files:

1. For each data source, copy the IDS or ADS configuration file from the following Isentris installation:

   Windows:

   `{Isentris installation directory}\system\config\datasource\dsfiles`

   to the Pipeline Pilot Server installation:

   `<pps_install>\web\queryservice\dsfiles`

   Linux:

   `{Isentris installation directory}/system/config/datasource/dsfiles`

   to the Pipeline Pilot Server installation:

   `<pps_install>/web/queryservice/dsfiles`

   Note: The Isentris server may be on a different computer from BIOVIA Pipeline Pilot Server.

   IMPORTANT! After you copy the IDS or ADS configuration file into the `dsfiles` folder, it is published. You MUST test and validate the IDS or ADS before you allow users to have general access to it.

2. For each IDS or ADS, configure a corresponding relational data source:
   i. In a text editor, open the Isentris relational data source XML file that corresponds to the IDS or ADS. The name of the relational data source is indicated by the `sourceName` attribute of the `ConnectionAttributes` element in the IDS file.
   ii. Create a relational data source in Pipeline Pilot Server using the settings in the Isentris XML relational data source file. For instructions on doing this, see Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal.

IMPORTANT!

- Do not give the relational data source that you create in Pipeline Pilot Server the same name as an XML relational data source in the \<pps_install>\web\queryservice\dsfiles folder. This is because the presence of the XML relational data source file would override any access restrictions on the Pipeline Pilot Server data source, and create a potential security loophole.
- Although it is possible to publish the existing XML relational data source by copying its file to the \<pps_install>\web\queryservice\dsfiles folder (see above), you **should not do this**, because the data source would then circumvent BIOVIA Foundation security. You should create a relational data source in the Pipeline Pilot Admin Portal.

After republishing the Isentris data source files, you can configure the Query Service catalog.

Creating a Data Source XML File Manually

To create a data source using the template data source configuration files:

1. Navigate to this location:

   Windows:

   ```
   <pps_install>\apps\scitegic\queryservice\isentris\system\config\datasource\templates\ids
   ```

   Linux:

   ```
   <pps_install>/apps/scitegic/queryservice/isentris/system/config/datasource/templates/ids
   ```

2. Find an IDS or ADS configuration file that is suitable for the new data source and copy it to this location:

   Windows:

   ```
   <pps_install>\apps\scitegic\queryservice\dsfiles
   ```

   Linux:

   ```
   <pps_install>/apps/scitegic/queryservice/dsfiles
   ```

3. Configure a relational data source for the database in the Pipeline Pilot Admin Portal, as described in Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal. The following settings are mandatory:

Field	Value
Name	The name of this relational data source. This must be the name specified in the IDS file, in the `sourceName` attribute of the `ConnectionAttributes` element (for example, `<ConnectionAttributes sourceName="MyChemDB_REL" />`).

Field	Value
	IMPORTANT! Ensure that the name is not already being used by an XML relational data source published in the folder `<pps_install>\web\queryservice\dsfiles` Such an XML relational data source file might have been created for use with BIOVIA Isentris or with early versions of the BIOVIA Query Service. If a published XML relational data source has the same name as a Pipeline Pilot Server relational data source, its presence overrides any access restrictions on the Pipeline Pilot Server data source. This situation creates a potential security loophole, and should be avoided. For example: ■ Choose a different name for the new relational data source. For example, if the `dsfiles` folder contains an old Isentris relational data source file called `ACD_REL.xml`, do not name the new data source `ACD_REL`, but use an alternative such as `ACD_PPS_REL`. ■ Verify whether the XML relational data source is still needed, and if it is not, delete it. Note that XML RDS files are deprecated. You should create new relational data sources in Pipeline Pilot Server.
Type	`JDBC`

Driver	**Oracle database:**
	`Oracle.jdbc.OracleDriver`
	SQL Server database:
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`

Connection String	The database connection string. Use a string with the appropriate format for your database:
	Oracle database using service name:
	`jdbc:oracle:thin:@<server>:<port e.g. 1521>/<service_name>`
	Oracle database using service ID:
	`jdbc:oracle:thin:@<server>:<port e.g. 1521>:<service_id>`
	SQL Server database:
	`jdbc:sqlserver://<server_name>:<port e.g. 1433>;databaseName=<database_name>`
	The correct format of the string is indicated by the template below the text box. You may find it helpful to copy and paste the string, inserting appropriate values to replace the values in angled brackets. Your database administrator can provide you with the correct details.
	If you use the text in the template, make sure you remove all angled brackets.
	Examples:
	`jdbc:Oracle:thin:@MyServer:1521/MyServiceName` (Oracle with

Field	Value
	service name) `jdbc:Oracle:thin:@MyServer:1521:MySID` (Oracle with SID) `jdbc:sqlserver://myServer:1433;databaseName=myDatabase` (SQL Server)
Optional DB Username	THIS IS REQUIRED. The user name for the Oracle or SQL Server database. Consult your database administrator.
Optional DB Password	THIS IS REQUIRED. The password for the Oracle or SQL Server database.

Configuring an IDS to Support Retrieval of Molecules As Pipeline Pilot Chemistry Instead of CHIME Strings

(**Optional**) To enable the IDS to support retrieval of molecules as Pipeline Pilot Chemistry instead of CHIME strings and facilitate a possible performance advantage, modify the IDS by performing the steps that follow. This is not a required procedure, but it might improve performance.

Notes:

- Perform these steps for any IDS that uses molecules in Pipeline Pilot Chemistry format from BIOVIA Direct 9.
- Searching of chemistry data is possible with Oracle databases that have the BIOVIA Direct cartridge installed. Chemistry searching is not available for SQL Server databases. Chemistry data can, however, always be retrieved in results.

To modify your IDS file to support retrieval of molecules as Pipeline Pilot Chemistry:

1. Navigate to the folder containing the IDS file you want to modify.
2. Use a text editor to open the file.
3. Do a search and find the CTAB element(s).
4. Set the `isHidden` attribute to `false`:
   `isHidden="false"`
5. Set the `outputFormat` attribute of the CTAB element to `PPSerializedMolecule`:
   `outputFormat="PPSerializedMolecule"`
   If this attribute is not present, add it.
6. Search for any other CTAB elements and modify them.
7. Save and close the file.

Example

Below are excerpts of sample IDS files, the first using an underlying Oracle database, the second a SQL Server database.

Oracle:

```
<Connectors>
  <Connector name="Mol" id="MOLroot"
             type="OneToManyDataConnector"
             toTable="&MOLSCHEMA;&MOLSCHEMA2D;_MOLTABLE"
             validation="fail">
```

```
    <OutputFields>
    .
    .
    .
    <OutputField name="CTAB" isHidden="false" type="Structure"
                 molFormat="Chime" display_mode="Q"
                 outputFormat="PPSerializedMolecule"
                 ISINDEXED="true" LENGTH="4000" />
    .
    .
    .
    </OutputFields>
  </Connector>
</Connectors>
```

SQL Server:

```
<Connectors>
  <Connector name="Mol" id="MOLroot"
             type="OneToManyDataConnector"
             toTable="&MOLSCHEMA;&MOLSCHEMA2D;_MOLTABLE"
             validation="fail">
    <OutputFields>
    .
    .
    .
    <OutputField name="CTAB" type="Structure"
                 outputFormat="PPSerializedMolecule"
                 display_name="Molstructure"
                 SORTRESTRICTION="SORT_NOSORT"
                 searchable="false" />
    .
    .
    .
    </OutputFields>
  </Connector>
</Connectors>
```

Configuring Query Service in the Administration Portal

In the Pipeline Pilot Admin Portal, you can configure properties for the Query Service.

To set the Query Service properties:

1. Go to the BIOVIA Pipeline Pilot Server home page, and click **Administration Portal**.
2. If prompted, enter the user name and password of a Pipeline Pilot Server administrator.
3. Select **Setup > Global Properties**.
4. From the **Package** dropdown list, select **BIOVIA/Query Service**.
5. Set the following properties:
   - **logLevel** - Level of error logging:
     - ERROR
     - DEBUG
     - INFO
     - TRACE
     - WARN

- OFF
- ALL

- **timeOut** - Controls the timeout used for all Query Service calls. The unit of measurement is seconds. The default is 300 seconds.

Configuring the Query Service Catalog

The Query Service catalog contains predefined query and data transformation definitions that can be executed using Query Service commands. It is useful as a repository for actions that are frequently performed, since it removes the need to specify a complete query or transform every time it is executed using HTTP.

The catalog is defined in the file `Catalog.xml`, which is located in the following location:

Windows:

```
<pps_install>\apps\scitegic\queryservice
```

Linux:

```
<pps_install>/apps/scitegic/queryservice
```

You can add a query to the catalog in one of these ways:

- Editing the `Catolog.xml` file
- Submitting an HTTP POST request on `catalog/queries` -- For information about how to do this, see the *Query Service Developer's Guide*.

Editing the Catalog.xml File

The more direct way to define a new query in the Query Service catalog is to edit the `Catalog.xml` file. You can modify an existing definition, clone and edit an existing definition, or create a new definition from scratch.

IMPORTANT! You are strongly advised to create a backup copy of `Catalog.xml` before you edit it.

If you want to modify or clone a query definition, you may find it easiest to use a simple query definition as your basis, such as `CDBRegnoExact` in the ACD IDS (ACS_IDS).

Using the definition as a template, you can modify any of the elements or attributes listed below:

XML Element or attribute to change (XPath syntax)	Description
`@name`	The name of the query. This is used to reference the query in a URL.
`@datasource`	The name of the data source to query. This must be an IDS or an ADS. If you have an existing Isentris data source, change this value to the name of your Isentris data source.
`@description`	Optional query description.

XML Element or attribute to change (XPath syntax)	Description
`QueryString`	The query to execute. This must be in Unified Query Language (UQL) syntax. Use question marks ? to specify query parameters. These correspond sequentially to the parameters defined in `QueryParam` elements. **Note:** CDATA notation is optional: alternatively, you can specify the query as a simple XML text node.
`QueryParams`	If the query has parameters, specify each parameter in a `QueryParam` child element. If the query does not have parameters, remove this element or leave it empty.
`QueryParams/QueryParam`	Contains the name, type, and description of a query parameter. The parameter must be represented as ? in the query string. Parameters must be defined in the correct sequence.
`Properties`	Contains Isentris-related properties. If you are not using an Isentris data source or do not need to set an Isentris property, remove this element.
`Transform`	Contains Isentris-related transform definitions. If you are not using an Isentris data source or do not need to perform data transformation, remove this element.

Checking the Query Service Catalog

To verify that the Query Service is configured properly and that the catalog is in place, submit the following URL on your browser:

```
http://{host}:{port}/auth/queryservice/catalog/queries
```

For example:

```
http://vm-pps-1234:9944/auth/queryservice/catalog/queries
```

The response returns the queries, transforms, and histories that are defined in the catalog.

Chapter 4:
Optional Configuration Tasks

The following are optional configuration tasks:

- Modifying the Definition of the Automatic Structure Search
- Modifying Exact Structure Query Type Searches
- Setting up Custom .jar Files and Configuration Files
- Setting Properties for Shared Data Sources

For information about required configuration tasks, see Required Configuration.

Modifying the Definition of the Automatic Structure Search

The automatic structure search enables the user to perform a search for a reaction or molecule structure query consisting of a sequence of individual searches; the sequence of searches will continue automatically and without user intervention until a search returns one or more hits, or the last search in the sequence has been executed. The sequence of searches is defined in the `querymetadata.xml` file, located at the following location:

Windows:

`<pps_install>\web\queryservice\isentris\system\config\datasource`

Linux:

`<pps_install>/web/queryservice/isentris/system/config/datasource`

IMPORTANT! Be sure to back up the `querymetadata.xml` file before you edit it.

The default sequence provided by the Query Service includes Exact, Flexmatch, Similarity, Substructure and, in the case of a reaction query, Transformation searches. The sequence is ordered in such a way that the search proceeds from the specific to the more general. The definition of each search, and the sequence in which they are executed, can be changed by editing this file.

The automatic structure search for a reaction query is defined by the function `autoReaction`, and for a molecule query by the function `autoStructure`. In each case, the name of the search that is displayed in the client application can be changed by editing the `alias` property; by default these are named **Automatic Reaction Search** and **Automatic Molecule Search** respectively.

You can modify only the values shown in bold type in the following example. You also can remove any of the `Alternate` elements by commenting them out or deleting them from the code.

IMPORTANT! You must remove the entire section or the XML file will not work.

Note: Before changing the definition of individual search types, see the *BIOVIA Direct Administration Guide*.

Example

This code example shows the `autoReaction` parameters in the `querymetadata.xml` file.

```
<AutomaticQuery functionName="autoReaction">
  <ParameterSub>reactionField</ParameterSub>
  <ParameterSub>chimeString</ParameterSub>
  <ParameterSub>ancillaryTag</ParameterSub>
```

```
<ParameterSub>narrowClassificationField</ParameterSub>
<ParameterSub>narrowClassification</ParameterSub>
<ParameterSub>mediumClassificationField</ParameterSub>
<ParameterSub>mediumClassification</ParameterSub>
<ParameterSub>broadClassificationField</ParameterSub>
<ParameterSub>broadClassification</ParameterSub>
  <Alternate description="Exact" functionName="RxnFlexMatch">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>'match=fra,sal,ion,cha,hyd,mas,rad,val,bon,met,tau,ste,mix,
      pol,typ,end,msu,dat'</Parameter>
  </Alternate>
  <Alternate description="Narrow Classification">
    <Parameter>narrowClassificationField</Parameter>
    <Parameter>narrowClassification</Parameter>
  </Alternate>
  <Alternate description="Medium Classification">
    <Parameter>mediumClassificationField</Parameter>
    <Parameter>mediumClassification</Parameter>
  </Alternate>
  <Alternate description="Broad Classification">
    <Parameter>broadClassificationField</Parameter>
    <Parameter>broadClassification</Parameter>
  </Alternate>
  <Alternate description="Substructure" functionName="RSS">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 70/20" functionName="RxnSim">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>’70 20’</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 60/20" functionName="RxnSim">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>’60 20’</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 50/20" functionName="RxnSim">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>’50 20’</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 40/20" functionName="RxnSim">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>’40 20’</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 30/20" functionName="RxnSim">
    <Parameter>reactionField</Parameter>
    <Parameter>chimeString</Parameter>
    <Parameter>’30 20’</Parameter>
    <Parameter>ancillaryTag</Parameter>
  </Alternate>
  <Alternate description="Similarity 20/20" functionName="RxnSim">
```

```
      <Parameter>reactionField</Parameter>
      <Parameter>chimeString</Parameter>
      <Parameter>'20 20'</Parameter>
      <Parameter>ancillaryTag</Parameter>
    </Alternate>
  </AutomaticQuery>
</Function>
```

Modifying Exact Structure "Query Type" Searches

Using Flexmatch searches you can add custom search types for all your users.

If you want to add, modify, or comment out 'exact' Flexmatch query types, you can edit the `querymetadata.xml` file. This file is at the following location:

Windows:

`<pps_install>\web\queryservice\isentris\system\config\datasource`

Linux:

`<pps_install>/web/queryservice/isentris/system/config/datasource`

Note: Modifying this file is a global change, and you must be careful that you use the correct Flexmatch switches in the XML. For detaied information on Flexmatch switches, see the *Chemical Representation guide*.

To Modify the XML File

IMPORTANT! Back up the `querymetadata.xml` file before you modify it.

1. Open the `querymetadata.xml` file in any XML editor.
2. Locate the Flexmatch query type you want to modify, for example:

```
<Function alias="Exact" name="FlexMatch" type="Integer"
          fixedComparison="=1" description="flexmatch search"
          location="where">
  <Args>
    <Arg name="CTAB" type="_Field"
         description="structure field"/>
    <Arg name="query" type="Structure" valueNeedsQuote="true"
         description="structure query"/>
    <Arg isConstant="true" value="'match=fra,sal,ion,cha,hyd,
mas,rad,val,bon,met,tau,ste,mix,pol,typ,end,msu,dat'"/>
  </Args>
</Function>
```

3. Make your changes.
4. Save the file under the same name, and close the editor.

Note: Before you install an upgrade for Pipeline Pilot Server or the Query Service, you must move this file to another location to keep the changes you have made. After the upgrade is complete, place the file back in its original location.

Enabling Matching of Total Charge in Substructure Mapping of Pi Systems

In BIOVIA Direct 9.1 and later, the flag `IgnoreChargesInPiSystems` exists, and the Query Service applies it by default to searches that use the chemistry operators `sss`, `flexmatch`, `rss`, and `rxnflexmatch`. This flag causes total charge to be ignored in substructure mappings of pi systems. If you want total charge to be considered in these searches, you must edit the `querymetadata.xml` file. This file is at the following location:

Windows:

`<pps_install>\web\queryservice\isentris\system\config\datasource`

Linux:

`<pps_install>/web/queryservice/isentris/system/config/datasource`

IMPORTANT! Modifying this file is a global change, so be very careful when making edits, and ALWAYS back up the old file.

Note: The `IgnoreChargesInPiSystems` flag is always applied by the Flexmatch search in the Query Builder in the Query Service Component Collection, because it is hard-coded there.

To Modify the XML File

1. Make a backup copy of `querymetadata.xml`.
2. Open the `querymetadata.xml` file in any XML editor.
3. Remove the `IgnoreChargesInPiSystems` flag from the specification of the function from which you want to exclude it. The `IgnoreChargesInPiSystems` flag is specified in an `Arg` element with the attribute `isConstant="true"`. Remove the flag as follows:
   - If `IgnoreChargesInPiSystems` is the only flag, remove the whole `Arg` element. Example:

```
<Function alias="Substructure" name="SSS" type="Integer" fixedComparison="=1"
          description="substructure search" location="where">
  <Args>
    <Arg name="CTAB" type="_Field" description="structure field"/>
    <Arg name="query" type="Structure" valueNeedsQuote="true"
         description="structure query"/>
    <Arg isConstant="true" value="'IgnoreChargesInPiSystems'"/>
    <Arg name="sssNumber" type="_AncillaryTag"
         description="used in ancillary functions"/>
  </Args>
</Function>
```

   - If `IgnoreChargesInPiSystems` belongs to a set of flags, remove just `IgnoreChargesInPiSystems` from the `Arg/@value` attribute. Example:

```
<Function alias="Exact" name="FlexMatch" type="Integer" fixedComparison="=1"
description="flexmatch search" location="where">
  <Args>
    <Arg name="CTAB" type="_Field" description="structure field"/>
    <Arg name="query" type="Structure" valueNeedsQuote="true"
description="structure query"/>
    <Arg isConstant="true" value="'IgnoreChargesInPiSystems
match=fra,sal,ion,cha,hyd,mas,rad,val,bon,met,tau,ste,mix,pol,typ,end,msu,dat'"/>
  </Args>
</Function>
```

4. Save `querymetadata.xml`.

Note: Before you install an upgrade for Pipeline Pilot Server or the Query Service, you must move this file to another location to keep the changes you have made. After the upgrade is complete, place the file back in its original location.

Setting Up Custom JAR Files and Configuration Files

To make custom classes and configurations available to the Query Service, you must manually add custom JAR file(s) or configuration files to specific locations. After you have added them manually, they are loaded automatically.

To set up a custom .jar file:

1. Copy the custom JAR file into this location:

   Windows:

   `<pps_install>\web\queryservice\jars`

   Linux:

   `<pps_install>/web/queryservice/jars`

   Note: You might have to create the `jars` folder or directory yourself.

2. On the Pipeline Pilot Server host computer, click **Start**, and then select **Control Panel > Administrative Tools > Services**.
3. Right-click **BIOVIA Pipeline Pilot {version number} (Tomcat)** and select **Restart**.

To set up a custom configuration file:

Note: Custom configuration files are stored in subfolders of the `jars` folders in the preceding steps. These files can be referenced relative to the `jars` folder or directory in the custom classes.

1. Copy the custom configuration file into this location:

   Windows:

   `<pps_install>\web\queryservide\jars\conf`

   Linux:

   `<pps_install>/web/queryservice/jars/conf`

2. On the Pipeline Pilot Server host computer, click **Start** and then select **Administrative Tools > Services**.
3. Right-click **BIOVIA Pipeline Pilot {version number} (Tomcat)** and select **Restart**.

Setting Properties for Shared Data Sources

The following properties are used when you configure shared data sources:

- SHARED
- STATIC_CATALOG_METADATA

You can set these properties in Data Source Builder or use a text editor and manually edit the XML file for the data source. For more information about using the Data Source Builder, see the online help for Data Source Builder. For more information about editing the XML file for the data source, see the *Query Service Developer's Guide*, the *Query Service Integrating Data Source Guide*, and the *Query Service Data Source XML Reference Guide*.

To share a data source:

- Set SHARED and STATIC_CATALOG_METADATA to TRUE.

Chapter 5:
Setting Up the Query Service Examples

Setting Up Example Protocols with the Research Workbench (RWB) Database

So that end users can use the Query Service example protocols with the Research Workbench (RWB) database, you must perform these steps:

1. Follow the procedure in Installing the Research Workbench (RWB) Database.
2. To test them in the Pipeline Pilot Client, open the **Protocols** tab, and expand the **Examples** folder.

Setting Up Example Protocols with the ACD Database

1. Verify that you have a license to the ACD database, and that it is properly installed.
2. Verify that the ACD _rel and ACD _ids configuration files are published, tested, and validated in Data Source Builder.

   Note: The example protocols are written to use a data source called ACD. If you publish the ACD IDS with a different name, the examples do not run successfully.
3. Add the relational data source using the Pipeline Pilot Admin Portal. For more information, see Configuring Query Service Data Sources.
4. Copy the IDS file from the templates folder to the `dsfiles` folder:

   Windows:

   `<pps_install>\web\queryservice\isentris\system\config\datasource\templates\ids`

   to this location:

   `<pps_install>\web\queryservice\dsfiles`

   Linux:

   `<pps_install>/web/queryservice/isentris/system/config/datasource/templates/ids`

   to this location:

   `<pps_install>/web/queryservice/dsfiles`
5. To test the protocols in the Pipeline Pilot Client, open the **Protocols** tab, and expand the **Examples** folder.

Setting Permissions for Using Web Port to Run Example Protocols

Typically, all BIOVIA Pipeline Pilot Server users are part of the group WebPort/Users. This means that all users have access to Web Port and can run the Web Port example protocols from Web Port. If a user is having trouble running the examples from Web Port, verify that they have their permissions set correctly. To verify, open the Pipeline Pilot Admin Portal and navigate to **Security > Groups**.

Chapter 6:
Troubleshooting

This chapter provides troubleshooting information.

Using the Query Service Log File

Permissions: You must log in as a your operating system's Administrator.

The Query Service log is specific to Query Service; only the Query Service writes to this log. If you are having problems with the Query Service, try checking its log first:

1. Navigate to the following location:

   Windows:

   `<pps_install>\logs\messages\javaserver_queryService.log`

   Linux:

   `<pps_install>/logs/messages/javaserver_queryService.log`
2. Open `javaserver_queryService.log` with a text editor and review any entries.
3. Close the file.
4. If there was no relevant information, check the Tomcat log file next, as described in Using the Tomcat Log File on Windows and Using the Tomcat Log File on Linux.

Using the Tomcat Log File on Windows

Permissions: You must be logged in as your operating system's Administrator.

If you are having problems with the Query Service, try the following procedure:

1. On the computer that hosts BIOVIA Pipeline Pilot Server, click **Start**, and then select **Administrative Tools** > **Component Services**.
2. Click **Services**.
3. Right-click **BIOVIA Pipeline Pilot {version number} (Tomcat)** and select **Stop**.
4. Navigate to the `javaserver_tomcat.log` file at the following location:

   `<pps_install>\logs\messages`
5. Rename the `javaserver_tomcat.log` file to `javaserver_tomcat.log.backup`.
6. Go back to **Services**.
7. Right-click **BIOVIA Pipeline Pilot {version number} (Tomcat)**, and select **Start**.
8. With a text editor, open the new `javaserver_tomcat.log` file.
9. Search for the string `query service` and you should find some information that helps you troubleshoot the problem.
10. When you are finished, close the file.

Using the Tomcat Log File on Linux

Permissions: You must log in as your operating system's Administrator.

If you are having problems with the Query Service, try the following procedure:

1. Change directories to your server install directory:
   ```
   > cd linux_bin
   ```
2. Stop the Linux server:
   ```
   >./stopserver
   ```
3. Ensure that the ports are released:
   ```
   > netstat -lt
   ```
   A list of port numbers currently in use is displayed. The default Apache port numbers used by BIOVIA Pipeline Pilot Server should not be listed.

   Tips:
   - It might take a few minutes for Apache to release the ports. To refresh the list, retype the command.
   - You can also use the following command to check for running HTTPD processes in the directory:
     ```
     > ps -ef | grep httpd
     ```
4. Navigate to the `javaserver_tomcat.log` file at the following location:
   ```
   <pps_install>/logs/messages.
   ```
5. Rename the `javaserver_tomcat.log` file to `javaserver_tomcat.log.backup`.
6. Manually start the Linux server:

   IMPORTANT! Perform this task only if you did not create a boot script to automatically start BIOVIA Pipeline Pilot Server when you installed the application (this is not recommended). For further information about boot scripts, see the *BIOVIA Foundation Hub Installation and Configuration Guide* and the *BIOVIA Pipeline Pilot Server Admin Portal Guide*.

   a. Change the working directory to the program directory for your BIOVIA Query Service installation:
      ```
      > cd <pps_install>/linux_bin
      ```
   b. Start the server:
      ```
      > ./startserver
      ```
7. With a text editor, open the new `javaserver_tomcat.log` file.
8. Search for the string `query service` and you should find some information that helps you troubleshoot the problem.
9. When you are finished, close the file.

Using the Monitor REST Endpoints

The monitor REST endpoints return the status of various resources, and can be helpful in troubleshooting problems with the Query Service:

- `monitor/cache`: Provides information about what is currently in the Query Service cache. A DELETE request on this resource can be used to remove objects from the cache.
- `monitor/health`: Provides information about the status of the Query Service.
- `monitor/system`: Retrieves system information about Query Service in XML format.

Examples

```
http://localhost:9944/auth/queryservice/monitor/cache
```

```
http://localhost:9944/auth/queryservice/monitor/health
```

```
http://localhost:9944/auth/queryservice/monitor/system
```

For more information, see the *BIOVIA Query Service Developer's Guide*.

Appendix A:
Installing the RWB Database

BIOVIA Foundation is supplied with data sources for the Research Work Bench (RWB) database on Oracle.

If you have BIOVIA Direct cartridge installed in Oracle, you can set up the RWB database for searching of chemistry data. The RWB database is supported for Direct 8 and higher.

If your Oracle instance does not have BIOVIA Direct installed, you cannot search chemistry data, but you can retrieve it.

Note: These instructions are for Oracle only. BIOVIA does not provide data sources for the RWB database on SQL Server.

Permissions: To perform the following steps, you must have DBA permissions and be an Administrator on your operating system.

To install the RWB database:

1. Create an Oracle user as the owner of the RWB data.
   i. (**Optional**) If you need to create a new table space for RWB, you can use a command like the following one:

```
CREATE TABLESPACE "RWB_DATA"
       DATAFILE '<oracle_installation>\Oracle\oradata\my_db\rwb_data01.dbf'
       SIZE 256M AUTOEXTEND ON NEXT 64M EXTENT MANAGEMENT LOCAL;
```

   Note: Please consult the Oracle documentation, as your operating system and Oracle installation might require a slightly different syntax to be used.

   ii. Log in to Oracle as a database administrator, and run the commands below. The user name and passwords in the statements below are the default values, and can be modified if required. Adjust the tablespace names according to your Oracle instance. About 250 megabytes of tablespace are required for the RWB data.

```
CREATE USER "RWB_OWNER" PROFILE "DEFAULT" IDENTIFIED BY "RWB_OWNER"
DEFAULT TABLESPACE "RWB_DATA" TEMPORARY TABLESPACE "TEMP" ACCOUNT
UNLOCK;

ALTER USER "RWB_OWNER" QUOTA UNLIMITED ON RWB_DATA;

GRANT CONNECT TO "RWB_OWNER";

GRANT CREATE SYNONYM TO "RWB_OWNER";

GRANT CREATE SEQUENCE TO "RWB_OWNER";

GRANT CREATE TABLE TO "RWB_OWNER";

GRANT CREATE TRIGGER TO "RWB_OWNER";

GRANT CREATE VIEW TO "RWB_OWNER";
```

2. Import the Oracle data pump dump file `rwb.dmp` into the schema you just created.

   The RWB data is provided as an Oracle data pump dump file. To load that file into Oracle, you need to specify an Oracle directory object and grant read/write access to it for the user you created

above. Here is a simple example of creating the directory and enabling it for the RWB_OWNER user, to be run from an account with DBA privileges:

```
CREATE OR REPLACE DIRECTORY dpump_dir AS '<pps_
install>\scitegic\queryservice\qsExampleData\data\RWB';
GRANT READ, WRITE ON DIRECTORY dpump_dir TO RWB_OWNER;
```

> **Note:** For details on Oracle data pump and Oracle directory objects, see the Oracle documentation, as your installation might require slightly different syntax to be used.

3. After the directory object is configured in Oracle, extract the content of the `rwb.zip` file to the folder assigned to the directory object. `<pps_install>\scitegic\queryservice\qsExampleData\data\RWB` contains the following files:
   - `rwb_install.txt`
   - `RWB.DMP`
   - `rwb_import.par`
   - `RWB_REL.xml`
   - `RWB.xml`
4. After the directory object is configured in Oracle, open the data pump import parameter file `rwb_import.par` in a text editor.
5. Modify the lines that specify the target Oracle schema and the target tablespace to match your user and tablespace as created previously.

   ```
   remap_schema=RWB_OWNER:your_Oracle_userremap_tablespace=RWB_DATA:your_
   target_tablespace
   ```
6. Save the file.
7. Open a command line and navigate to the Oracle directory folder to which you extracted the ZIP file.
8. Run the `impdp` data pump import utility to load the dump file:

   ```
   impdp RWB_OWNER/RWB_OWNER directory=dpump_dir parfile=rwb_import.par
   ```

   Once the import is done, the RWB database contains the following objects:

Tables

Name of table	Description
COMPOUND_MOLTABLE	Main molecule table
RESULTS_MAIN	Secondary assay results
WELL_MAIN	Primary assay results
PLATE_MAIN	Plate information
TBLRESULTRELATION	Relation between plates and results
TBLVARIABLES	Variables information
TBLEXPERIMENTS	Experiment information
TBLPROTOCOL	Protocol information

Views

Name of table	Description
RESULTS_MAIN_VW	Secondary results view
RESULTS_VW_STAT_VW	Aggregated secondary results view
WELL_MAIN_VW	Primary results view
WEL_VW_STAT_VW	Aggregated primary results view

9. Install the IDS definition file:
   i. Copy the file `RWB.xml` to the following location:
   Windows:
   `<pps_install>\web\queryservice\dsfiles\`
   Linux:
   `<pps_install>/web/queryservice/dsfiles/`
   ii. Edit the `<!ENTITY SCHEMA0 "RWB_OWNER.">` line near the top of the 'RWB.xml' file to match the owning Oracle user name.

   Note: A period `.` must be appended to the user name, as shown in the original entry.

10. Configure a relational data source for the RWB database in the Pipeline Pilot Admin Portal, as described in Adding and Configuring Relational Data Sources in the Pipeline Pilot Admin Portal. The following settings are mandatory:

Field	Value
Name	RWB_REL
Type	JDBC
Driver	Oracle.jdbc.OracleDriver

Connection String	The database connection string. Use a string with the appropriate format for your database:
	Oracle database using service name:
	`jdbc:oracle:thin:@<server>:<port e.g. 1521>/<service_name>`
	Oracle database using service ID:
	`jdbc:oracle:thin:@<server>:<port e.g. 1521>:<service_id>`
	The correct format of the string is indicated by the template below the text box. You may find it helpful to copy and paste the string, inserting appropriate values to replace the values in angled brackets. Your database administrator can provide you with the correct details.
	If you use the text in the template, make sure you remove all angled brackets.
	Examples:
	`jdbc:Oracle:thin:@MyServer:1521/MyServiceName` (Oracle with

Field	Value
	service name) `jdbc:Oracle:thin:@MyServer:1521:MySID` (Oracle with SID)
Optional DB Username	THIS IS REQUIRED. The user name for the Oracle database. Consult your database administrator.
Optional DB Password	THIS IS REQUIRED. The password for the Oracle database.

IMPORTANT!

- Do not give the relational data source that you create in Pipeline Pilot Server the same name as an XML relational data source in the `\<pps_install>\web\queryservice\dsfiles` folder. This is because the presence of the XML relational data source file would override any access restrictions on the Pipeline Pilot Server data source, and create a potential security loophole.
- Although it is possible to publish the RWB relational data source by copying its template XML file to the `\<pps_install>\web\queryservice\dsfiles` folder (see above), you **should not do this**, because the data source would then circumvent BIOVIA Foundation security. You should create a relational data source in the Pipeline Pilot Admin Portal.

11. Enable the Oracle user for BIOVIA Direct, and create a domain index on the moltable:
    i. Log in to Oracle as the user you specified above.
    ii. Execute one of the following sets of commands, according to your Direct version:

Direct version	Commands

8	`EXECUTE c$mdlichem80.mdlauxop.setup;`
	`CREATE INDEX compound_moltable_ix ON compound_moltable`
	`(CTAB) INDEXTYPE IS C$MDLICHEM80.MXIXMDL PARAMETERS`
	`('MOLECULES UNIQUE="MATCH=ALL" ');`

9	`EXECUTE c$direct90.mdlauxop.setup;`
	`CREATE INDEX compound_moltable_ix ON compound_moltable`
	`(CTAB) INDEXTYPE IS C$DIRECT90.MXIXMDL PARAMETERS`
	`('MOLECULES UNIQUE="MATCH=ALL" ');`

9.1	`EXECUTE c$direct91.mdlauxop.setup;`
	`CREATE INDEX compound_moltable_ix ON compound_moltable`
	`(CTAB) INDEXTYPE IS C$DIRECT91.MXIXMDL PARAMETERS`
	`('MOLECULES UNIQUE="MATCH=ALL" ');`

9.5	`EXECUTE c$direct95.mdlauxop.setup;`
	`CREATE INDEX compound_moltable_ix ON compound_moltable`
	`(CTAB) INDEXTYPE IS C$DIRECT95.MXIXMDL PARAMETERS`
	`('MOLECULES UNIQUE="MATCH=ALL" ');`

Direct version	Commands
2016	EXECUTE c$direct2016.mdlauxop.setup; CREATE INDEX compound_moltable_ix ON compound_moltable (CTAB) INDEXTYPE IS C$DIRECT2016.MXIXMDL PARAMETERS ('MOLECULES UNIQUE="MATCH=ALL" ');
2017	EXECUTE c$direct2017.mdlauxop.setup; CREATE INDEX compound_moltable_ix ON compound_moltable (CTAB) INDEXTYPE IS C$DIRECT2017.MXIXMDL PARAMETERS ('MOLECULES UNIQUE="MATCH=ALL" ');
2017 R2	EXECUTE c$direct2017r2.mdlauxop.setup; CREATE INDEX compound_moltable_ix ON compound_moltable (CTAB) INDEXTYPE IS C$DIRECT2017R2.MXIXMDL PARAMETERS ('MOLECULES UNIQUE="MATCH=ALL" ');
2018	EXECUTE c$direct2018.mdlauxop.setup; CREATE INDEX compound_moltable_ix ON compound_moltable (CTAB) INDEXTYPE IS C$DIRECT2018.MXIXMDL PARAMETERS ('MOLECULES UNIQUE="MATCH=ALL" ');

12. Test in one of the following ways:
    - As the RWB IDS in Data Source Builder.
    - Use it in Search under the name **Research Work Bench**.
    - Use the Query Service example protocols that specify RWB in Pipeline Pilot Client.

Appendix B:
Spanning Multiple Oracle or SQL Server Instances

BIOVIA Pipeline Pilot Server provides options for searching across multiple Oracle or SQL Server instances.

IMPORTANT!
If you copy sample code from this PDF document and paste it into a text editor, the text might change from the original. To ensure that the text is valid and usable, you might need to restore indents, remove extra spaces, and retype single and double quotation marks.

About Spanning Multiple Oracle or SQL Server Instances

If you want to span multiple Oracle or SQL Server instances, there are several options which are described in this appendix.

Note: If both local and remote searches are reasonably specific, almost any solution is adequate.

These are the options in order of preference for overall performance:

1. IDS over Oracle materialized views or SQL Server indexed views
2. IDS standard data source connector
3. IDS DBLink data source connector
4. IDS INCursor data source connector
5. IDS over Oracle or SQL Server view of DBLinked table

Some lower-ranked options do better in specific cases than higher-ranked options.

Note: You can choose to span multiple Oracle or SQL Server instances. However, the performance is always better when all the data is in the same Oracle or SQL Server instance.

IDS over Oracle Materialized Views or SQL Server Indexed Views

Oracle materialized views and SQL Server indexed views are solutions to handling queries spanning local and remote data. You create a DBLink to the remote data, and then create a materialized or indexed view in your local instance. The materialized or indexed view then gives you access to the remote data in the local instance by transparently creating a copy of the remote data in the local instance and synchronizing it with the remote table. Oracle and SQL Server provide a variety of methods for synchronizing the data that are specified at the time the view is created.

Advantages

- Fastest multiple-instance solution
- Very easy to implement because Oracle or SQL Server handles everything transparently
- Query performance not affected by WAN conditions between local and remote instances

Disadvantage

- Possibility of some performance issues if the remote data is very frequently updated. If the remote source is structural data, you no longer have access to the domain index. Hence structure searching

over a remote data source is very, very slow. This implies that if you want to combine searches across instances efficiently:

- Always use a local instance for your structure repository.
- Do not use multiple structure instances.

IDS Standard Data Source Connector

A separate data source using the remote Oracle or SQL Server instance is accessed using the DataSourceConnector. The connector executes and completes the search on the remote source, creates a local temporary table of these results, and then combines this table with the local search clauses.

This is easy to set up, and removes the datatype restriction present in the DBLink solution. The down side is that searches can be slow, because searches on the remote instance are driven to completion before all the link field values are downloaded to the local instance.

Advantage

- Works very well for simple lookups in remote sources.

Disadvantage

- Slow performance when the remote part of the search has structure criteria or it returns a lot of hits.

IDS DBLink Data Source Connector

A remote data source is accessed using the DataSourceConnector. Results are accessed using the DBLink. This option uses a data source connector with the `LIST_DBLINK` property specified. The connector executes and completes the search on the remote source, but creates a remote temporary table of the results. The connector combines this table with the local search clauses using the DBLink.

Advantage

- Faster than the standard data source connector for searches involving remote structure criteria.

Disadvantage

- Much slower than the standard data source connector for searches with non-specific remote data clauses.

IDS INCursor Data Source Connector

This is a data source connector with the `LIST_DBLINK` property specified and the `USE_INCURSOR` property set to `true`. This connector executes the search on the remote source without completing it initially, and returns first batch of hits. The default batch size is 100. This default size is editable by using the `MAXINCLAUSELITERALS` property, which cursors through the remote recordset combining each batch of results in turn with the local search criteria. If not enough hits are found, the next batch of remote hits is returned and the process continues.

This is a fully incremental solution that might not require local or remote searches to complete. However, if the remote search returns too many records but the local search is very specific, the total search time might be considerably longer than in the previous solution.

Advantage

- Faster than the standard data source connector or DBLink data source connector at returning the first few hits from a search in most cases.

Disadvantages

- Normally slower than the standard data source connector or DBLink data source connector for getting the entire recordset, for example when getting the record count.
- Unusably slow when the local part of the search is slow, for example in local structure searches where the remote data clause is non-specific.

When to Use

- Use when the remote data clause is possibly generic, and the local data clause is fast to evaluate.
- Use when you are only interested in the first few hits and you are likely to find them in the first batch evaluated.

Examples

Here are examples of how to set up an INCursor data source connector:

Create an IDS network for the remote data source

```
<DataSourcesByName>
  <DataSource NAME="REMOTEDATA_IDS" TYPE="IDS">
    <Connection>
      <ConnectionAttributes SOURCENAME="REMOTEDATA_REL"/>
      <Roots>

        <Root NAME="MOL"/>
      </Roots>
      <Connectors>
        <Connector NAME="MOL" TYPE="OneToManyDataConnector"
                   TOTABLE="“REMOTETABLE">
          <OutputFields>
            <OutputField NAME="CDBREGNO" isKey="true"/>
            <OutputField NAME="MDLNUMBER"/>
            <OutputField NAME="MOLNAME"/>
          </OutputFields>
        </Connector>
      </Connectors>
    </Connection>
  </DataSource>
</DataSourcesByName>
```

Create an IDS connector to connect the remote table

```
<Connector NAME="REMOTE_DATA" TYPE="“DatasourceConnector"
           FROMTABLE="MYLOCAL_MOLTABLE">
  <ConnectionAttributes SOURCENAME=" REMOTEDATA_IDS "/>
  <RootName>MOL</RootName>
  <LinkFields>
    <LinkField FROMFIELD="CDBREGNO" TOFIELD="CDBREGNO"/>
  </LinkFields>
  <Properties>
    <Property name="USE_NATIVE_METADATA" value="true"/>
```

```
    <Property name="USE_INCURSOR" value="true"/>
  </Properties>
</Connector>
```

IDS over Oracle or SQL Server View of DBLinked Table

All remote tables are accessed through individual DBLinks. You can set up a view or an alias of a remote table using a DBLink so that the IDS network treats it as a local table. This is much the same concept as the Oracle materialized view or the SQL Server indexed view, except that Oracle and SQL Server do not make local copies of the data. This requires no changes to a data network but, since Oracle and SQL Server do not optimize across links, search speeds can be slow.

Advantage

- Easy to set up.

Disadvantages

- The Oracle and SQL Server limitation means that you cannot have LOBs in the remote table.
- Performance is often very slow, because the Oracle and SQL Server optimizers:
  - Have no statistics on the remote table.
  - Often choose incorrect execution plans, as DBLinks are not intended for efficient querying and are designed primarily for replication purposes.