

IBM Cognos Analytics
Version 12.0.x

Notebook User Guide



©

Product Information

This document applies to IBM Cognos Analytics version 12.0.0 and may also apply to subsequent releases.

Copyright

Licensed Materials - Property of IBM

© Copyright IBM Corp. 2015, 2024.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft product screen shot(s) used with permission from Microsoft.

© **Copyright International Business Machines Corporation .**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Installing IBM Cognos Analytics for Jupyter Notebook Server.....	1
Hardware requirements for Jupyter Notebook Server.....	1
Installing Jupyter Notebook Server on Linux.....	2
Uninstalling Jupyter Notebook Server.....	4
Installing Jupyter Notebook Server on Microsoft Windows 10.....	4
Uninstalling Jupyter Notebook Server.....	6
Installing a pip package in an offline Linux environment.....	6
Installing a pip package in an offline Windows environment.....	7
Configuring Jupyter Notebook Server.....	8
Configuring the Cognos Analytics gateway for Jupyter Notebook Server.....	11
Notebook performance logs.....	11
Securing Jupyter Notebook Server.....	12
Upgrading IBM Cognos Analytics for Jupyter Notebook Server.....	13
Upgrading your installation for Linux.....	13
Upgrading your installation for Microsoft Windows.....	14
Upgrading Python packages and R packages.....	15
Adding additional Ubuntu operating system packages.....	16
Troubleshooting IBM Cognos Analytics for Jupyter Notebook Server.....	16
Chapter 2. Enabling IBM Cognos Analytics for Jupyter Notebook.....	17
Chapter 3. Getting started with Notebook.....	19
Creating a notebook.....	20
Notebook actions.....	20
Reading data from a data source.....	21
Writing data to a data source.....	25
Searching for data objects.....	26
File paths that contain a forward slash.....	28
Finding the ID of a file.....	28
Python notebook examples.....	29
R notebook examples.....	32
Uploading external notebooks.....	38
Importing Watson Studio notebooks to Cognos Analytics.....	39
Connecting Watson Studio notebooks to Cognos Analytics.....	40
Best practices for displaying notebook visualizations in dashboards.....	40
Chapter 4. Adding a Notebook widget.....	43
Chapter 5. Including output from a notebook.....	45
Chapter 6. Jupyter notebook samples.....	47
Flexible lightweight ETL notebook sample.....	47
Time series analysis notebook sample.....	47
Visualization creation notebook sample.....	47
Retailer dashboard notebook sample.....	47
Telecom data analysis notebook sample.....	47
Telecom data visualizations notebook sample.....	48
Unit infection data notebook sample.....	48
Data quality template notebook sample.....	48
Schedule data creation notebook sample.....	48

Schedule data creation notebook widget sample.....	48
Health insurance coverage analysis notebook sample	48
Retailer report notebook widget sample.....	48

Chapter 1. Installing IBM Cognos Analytics for Jupyter Notebook Server

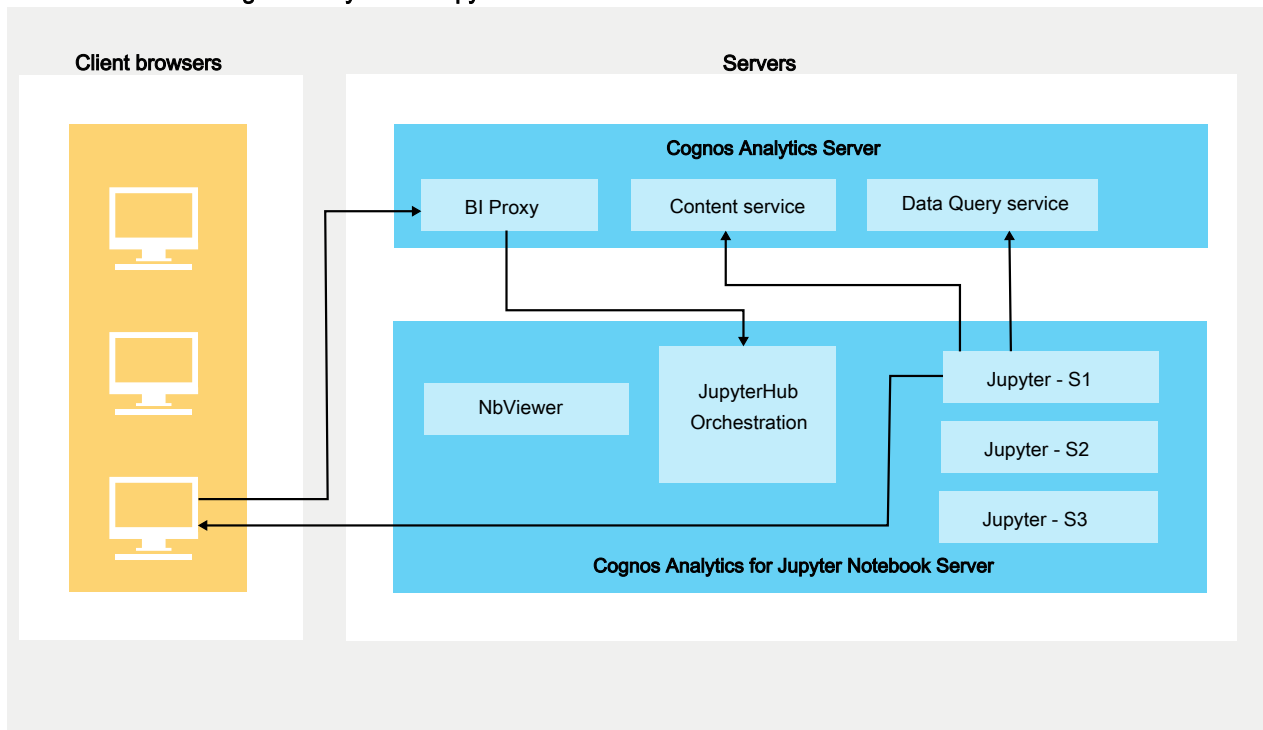
IBM® Cognos® Analytics includes a version of Jupyter Notebook that is available as a separate installer.

After you install Jupyter Notebook Server, Cognos Analytics users can create and edit Jupyter Notebook in Cognos Analytics.

The two main components of Jupyter Notebook Server are JupyterHub and the Notebook viewer (NbViewer). JupyterHub is the orchestrator that manages multiple server instances.

IBM Cognos Analytics for Jupyter Notebook Server can be installed on the same computer as IBM Cognos Analytics Server or on a different computer. The following diagram shows the server architecture.

Architecture of IBM Cognos Analytics for Jupyter Notebook



Before you install Cognos Analytics for Jupyter Server, determine your [hardware requirements](#).

After you install and configure IBM Cognos Analytics for Jupyter Notebook Server, the administrator must perform these tasks:

- Assign the Notebook capability to the appropriate users. For more information, see *Managing IBM Cognos Analytics*.
- Enable IBM Cognos Analytics for Jupyter Notebook. For more information, see *Managing IBM Cognos Analytics*.

Hardware requirements for Jupyter Notebook Server

The hardware requirements of a Cognos Analytics for Jupyter Server installation depend on the number of concurrent Notebook users and the complexity of their work.

To determine the sizing requirements of your Cognos Analytics Jupyter Server installation, consider these two factors:

- the number of active Notebook sessions that will run concurrently

- the complexity of the operations being performed in Notebook

This section provides some rough estimates of sizing examples.

Note: In the following sizing examples, the first three are for a single Cognos Analytics Jupyter Server. However, you can also deploy a clustered architecture behind a reverse-proxy, as described in this [high-failover example](https://community.ibm.com/community/user/businessanalytics/blogs/antonio-marziano/2019/06/18/setup-failover-jupyter-using-nginx-reverse-proxy) (<https://community.ibm.com/community/user/businessanalytics/blogs/antonio-marziano/2019/06/18/setup-failover-jupyter-using-nginx-reverse-proxy>).

Sizing example 1

- **Usage pattern:** For 10-20 data scientist users with approximately five concurrent Notebook edit/run/schedule sessions.
- **Sizing requirements:** Minimal.

Note: Notebook consumers, such as widgets in reports/dashboards, or users who view notebooks in read-only (nbViewer) mode do not contribute significantly to the resource requirements. Only Notebook users who edit, run, and schedule notebooks concurrently require increased resources.

Sizing example 2

- **Usage pattern:** For small-complexity usage patterns, such as data cleansing and visualizations.
- **Sizing requirements:**
 - Number of CPU Cores: 4
 - RAM: 16 Gb

Sizing example 3

- **Usage pattern:** For medium-complexity usage patterns, such as profile data quality, random forest classifier, KNN classification and decision trees.
- **Sizing requirements:**
 - Number of CPU Cores: 16
 - RAM: 64 Gb

Sizing example 4

- **Usage pattern:** For higher-complexity usage patterns, such as deep learning models or neural nets with pyTorch/Tensor Flow.
- **Sizing requirements:**
 - high complexity usage patterns have a different resource profile
 - For example, here are the minimum requirements for a [four-node IBM DSX-Local installation](https://www.ibm.com/support/knowledgecenter/SSAS34_1.2.1/local/requirements.html?view=kc#requirements__5node) (https://www.ibm.com/support/knowledgecenter/SSAS34_1.2.1/local/requirements.html?view=kc#requirements__5node).

Installing Jupyter Notebook Server on Linux

You can install IBM Cognos Analytics for Jupyter Notebook Server either on the same computer or on a different computer from where Cognos Analytics is installed.

The Jupyter Notebook Server supports Linux and Windows 10 platforms and requires Docker to be installed.

Note: Docker CE (Community Edition), Docker Engine, and Docker Desktop (CE) are supported at this time.

When you download and run the installer script, you load and start Docker containers. These containers allow Cognos Analytics users to create and edit Jupyter Notebook. By default, Cognos Analytics for

Jupyter Server is configured with many of the most common data science/analytic Python packages. In Cognos Analytics on Premises 11.1.2, Jupyter Server includes packages from PixieDust.

Tip: You can later [upgrade the Python packages](#) in your existing installation.

Before you begin

Before you install Jupyter Notebook Server, follow these steps:

1. Install Docker or Podman (on RHEL 8 or RHEL 9) as the container engine for Jupyter Notebook.

To install Docker:

Follow the procedures for one of these Linux distributions:

- [Installing Docker CE for CentOS](#)
- [Installing Docker CE for Ubuntu](#)
- [Installing Docker Engine for Red Hat Enterprise Linux](#)

You must be added to the Docker group for any Docker command to run without root privileges.

To install Podman on RHEL8 or RHEL9:

Follow the installation and configuration steps at [Podman](#).

Note: Throughout the Podman documentation, you will see Podman commands which correspond to particular topics where Docker commands were available.

One of Podman's greatest advantages is its complete CLI compatibility with Docker. When building Podman, Docker users can adapt without any significant changes. For example, you can use the alias command to create a docker alias for Podman:

```
yum install podman-docker
```

Important: To enable advanced networking features in Jupyter Server, you must also install the package podman-plugins. Type the following:

```
yum install -y podman-plugins
```

For more information on Podman, see this [Podman developer blog](#).

2. Set the fully qualified domain in Cognos Configuration.
 - a. In IBM Cognos Configuration, in the **Explorer** window, click **Environment**.
 - b. Under the **Dispatcher URI for external applications** set the fully qualified domain name (FQDN) for the IBM Cognos Analytics server.
 - c. Under the **Gateway URI** set the fully qualified domain name (FQDN) for the IBM Cognos Analytics server.
 - d. Click **File > Save**.
 - e. Restart the Cognos Analytics service.

About this task

For a demonstration of how to install Jupyter Notebook Server, [watch this video](#).

Procedure

1. Download the IBM Cognos Analytics for Jupyter Notebook Server installer and Server repository from [Passport Advantage](#).

Tip: See the Cognos Analytics 11.1.2 [Download Document](#) to find out which part number to download. Note that the Jupyter Server installer and Server repository are found only in the Linux eAssemblies.

2. Double click the installer file.
3. Follow the directions in the installation wizard to copy and install the files to your computer.

Tip: You can install on top of an older version of Jupyter Server.

The folder `jupyter_installation_location/dist` folder contains two subfolders:

- `dist/images`
- `dist/scripts`

Tip: The folder `dist/scripts/unix` contains all the scripts that you need to run.

Script	Purpose
<code>build.sh</code>	Run this to rebuild the images.
<code>config.conf</code>	Edit this configuration file to change Jupyter parameters.
<code>install.sh</code>	Run this to load and start the Docker containers.
<code>prune.sh</code>	Run this to remove old Docker images.
<code>start.sh</code>	Run this to start the Jupyter server.
<code>stop.sh</code>	Run this to stop the Jupyter server.
<code>uninstall.sh</code>	Run this to uninstall Jupyter server.

4. Ensure that you have execute permissions for each script:

Type `chmod -R u+x dist/scripts/unix`

5. Navigate to the `dist/scripts/unix` directory.

6. Type `./install.sh`

The install script runs.

Results

All of the Jupyter Server Docker images are loaded from the `jupyter_installation_location/dist/images` directory and the Docker containers are started.

What to do next

After installing IBM Cognos Analytics for Jupyter Notebook Server, the following tasks can be performed:

- If you want to change some default settings, you can [configure the Jupyter Notebook Server](#).
- The administrator must assign the Notebook capability to the appropriate users. For more information, see *Managing IBM Cognos Analytics*.
- The administrator must enable IBM Cognos Analytics for Jupyter Notebook. For more information, see *Managing IBM Cognos Analytics*.

Uninstalling Jupyter Notebook Server

To uninstall Jupyter Notebook Server, navigate to the `dist/scripts/unix` directory and then type `./uninstall.sh`

Installing Jupyter Notebook Server on Microsoft Windows 10

You can now install IBM Cognos Analytics for Jupyter Notebook Server for Microsoft Windows 10 either on the same computer or on a different computer from where Cognos Analytics is installed.

The Jupyter Notebook Server supports Linux and Microsoft Windows 10 platforms and requires Docker to be installed.

Note: Docker CE (Community Edition), Docker Engine, and Docker Desktop (CE) are supported at this time.

When you download and run the installer script, you load and start Docker containers. These containers allow Cognos Analytics users to create and edit Jupyter Notebook. By default, Cognos Analytics for Jupyter Server is configured with many of the most common data science/analytic Python packages. In Cognos Analytics on Premises 11.1.2+, Jupyter Server includes packages from versions of PixieDust.

Tip: You can later [upgrade the Python packages](#) in your existing installation.

Before you begin

Before you install Jupyter Notebook Server, follow these steps:

1. Verify that the **Hyper-V Platform** is running so that Linux containers can run on a windows host.
 - a. Open the **Control Panel**.
 - b. Click **Programs**.
 - c. Click **Turn Windows Features on or off**.
 - d. Find and expand the **Hyper-V** option.
 - e. Ensure that the **Hyper-V Platform** is checked.
2. Install Docker as the container engine for Jupyter Notebook.

Follow the Microsoft Windows 10 procedures at [Installing Docker Desktop for Windows](#)

Important: The Jupyter server requires that Docker use Linux containers. If your Docker Desktop installation was configured to use Windows containers, do one of the following tasks:

- Right-click the Docker icon in the bottom-right corner of the window and then select the option to switch to Linux containers.
- Re-install Docker Desktop, ensuring that you do not select the option to use Windows containers instead of Linux containers.

You must be added to the Docker group for any Docker command to run without root privileges.

3. Set the fully qualified domain in Cognos Configuration.
 - a. In IBM Cognos Configuration, in the **Explorer** window, click **Environment**.
 - b. Under the **Dispatcher URI for external applications** set the fully qualified domain name (FQDN) for the IBM Cognos Analytics server.
 - c. Under the **Gateway URI** set the fully qualified domain name (FQDN) for the IBM Cognos Analytics server.
 - d. Click **File > Save**.
 - e. Restart the Cognos Analytics service.

Procedure

1. Download the IBM Cognos Analytics for Jupyter Notebook Server installer and Server repository from [Passport Advantage](#).

Tip: See the Cognos Analytics 11.1.5 [Download Document](#) to find out which part number to download.

2. Double click the installer file.
3. Follow the directions in the installation wizard to copy and install the files to your computer.

Tip: You can install on top of an older version of Jupyter Server.

The folder `jupyter_installation_location/dist` folder contains two subfolders:

- `dist/images`
- `dist/scripts`

Tip: The folder `dist/scripts/windows` contains all the scripts and configuration files for windows installations.

Script	Purpose
build.bat	Run this to rebuild the images.
config.conf	Edit this configuration file to change Jupyter parameters.
install.bat	Run this to load and start the Docker containers.
prune.bat	Run this to remove old Docker images.
startup.bat	Run this to start the Jupyter server.
stop.bat	Run this to stop the Jupyter server.
uninstall.bat	Run this to uninstall Jupyter server.

4. Open a command prompt window with administrator privileges.
5. Navigate to the `dist/scripts/windows` directory.
6. Type `./install.bat`

The install script runs.

Results

All of the Jupyter Server Docker images are loaded from the `jupyter_installation_location/dist/images` directory and the Docker containers are started.

What to do next

After installing IBM Cognos Analytics for Jupyter Notebook Server, the following tasks can be performed:

- If you want to change some default settings, you can [configure the Jupyter Notebook Server](#).
- The administrator must assign the Notebook capability to the appropriate users. For more information, see *Managing IBM Cognos Analytics*.
- The administrator must enable IBM Cognos Analytics for Jupyter Notebook. For more information, see *Managing IBM Cognos Analytics*.

Uninstalling Jupyter Notebook Server

To uninstall Jupyter Notebook Server, open a command prompt window with administrator privileges and navigate to the `dist/scripts/windows` directory and then type `./uninstall.bat`

Installing a pip package in an offline Linux environment

Install Jupyter Notebook Server, including the additional pip package, without internet access.

About this task

When installing Jupyter Notebook Server, you need to install the PixieDust package `additional_pip_packages.txt`. This task requires internet access, which in some cases might not be available. So you need to download the package before installing Jupyter.

Procedure

1. Locate the `tar.gz` file for your specific package online, and download it.

The Python Package Index (PyPI) pypi.org (<https://pypi.org>) website can be used to download most pip packages from the source.

2. Navigate to the `/opt/ibm/cognos/jupyter/dist/scripts/` directory, and create a new directory named `tmp`.

3. Place all of the tar.gz packages that you downloaded into the tmp directory.
4. Open the file /opt/ibm/cognos/jupyter/dist/scripts/Dockerfile_server_instance for editing.
5. Modify the file in the following way:
 - a) Under the line:

```
COPY additional_pip_packages.txt /home/ca_user
```

Add the following new line:

```
COPY tmp/ /tmp/
```

This line instructs Docker to take your packages and place them into the Docker container during the build.

- b) Comment out the following section:

```
#COPY additional_conda_packages.txt .
#RUN if [ -s additional_conda_packages.txt ]; then \
# conda install --yes --file additional_conda_packages.txt; \
# fi \
#&& rm additional_conda_packages.txt
```

6. Save the Dockerfile_server_instance file ensuring that it's saved without a file extension.
7. Open the file /opt/ibm/cognos/jupyter/dist/scripts/additional_pip_packages.txt for editing.
8. Modify the file in the following way:
 - a) Remove the line pixiedust==1.1.17
 - b) Add the following new line /tmp/<package-name>.tar.gz. Ensure that the path matches the exact name of your tar.gz file.
 - c) Add a new line for every package that you want to install this way.
9. Save the additional_pip_packages.txt file.
10. Run the Linux installation script by using the following command: /opt/ibm/cognos/jupyter/dist/scripts/unix/install.sh.

Installing a pip package in an offline Windows environment

Install Jupyter Notebook Server, including the additional pip package, without internet access.

About this task

When installing Jupyter Notebook Server, you need to install the PixieDust package additional_pip_packages.txt. This task requires internet access, which in some cases might not be available. So you need to download the package before installing Jupyter.

Procedure

1. Locate the tar.gz file for your specific package online, and download it.

The Python Package Index (PyPI) pypi.org (<https://pypi.org>) website can be used to download most pip packages from the source.
2. Navigate to the C:\Program Files\ibm\cognos\jupyter\dist\scripts directory, and create a new directory named tmp.
3. Place all of the tar.gz packages that you downloaded into the tmp directory.
4. Open the file C:\Program Files\ibm\cognos\jupyter\dist\scripts\Dockerfile_server_instance for editing.
5. Modify the file in the following way:

a) Under the line

```
COPY additional_pip_packages.txt /home/ca_user
```

add the following new line

```
COPY tmp/ /tmp/
```

This line instructs Docker to take your packages, and place them into the Docker container during the build.

b) Comment out the following section:

```
#COPY additional_conda_packages.txt .
#RUN if [ -s additional_conda_packages.txt ]; then \
# conda install --yes --file additional_conda_packages.txt; \
# fi \
#&& rm additional_conda_packages.txt
```

6. Save the `Dockerfile_server_instance` file ensuring that it's saved without a file extension.
7. Open the file `C:\Program Files\ibm\cognos\jupyter\dist\scripts\additional_pip_packages.txt` for editing.
8. Modify the file in the following way:
 - a) Remove the line `pixiedust==1.1.17`
 - b) Add the following new line `/tmp/<package-name>.tar.gz`. Ensure that the path matches the exact name of your `tar.gz` file.
 - c) Add a new line for every package that you want to install this way.
9. Save the `additional_pip_packages.txt` file.
10. Run the Windows installation script by using the following command: `C:\Program Files\ibm\cognos\jupyter\dist\scripts\windows\install.bat`.

Configuring Jupyter Notebook Server

You can change the default settings in IBM Cognos Analytics for Jupyter Notebook Server by editing the `config.conf` file and by updating Cognos Configuration.

Procedure

1. Edit the `config.conf` file.
 - a) In a text editor, open the file `jupyter_installation_location/dist/scripts/unix/config.conf` file for Linux or the `jupyter_installation_location/dist/scripts/windows/config.conf` for Microsoft Windows 10.
 - b) Specify values, as required, for the parameters listed in the following table:

Parameter	Description
CERTIFICATES_DIRECTORY_PATH	<p>If you are securing Jupyter Notebook Server using SSL, enter the path to the directory that contains certificates for trusted SSL hosts.</p> <p>Tip: We recommend that the directory containing the certificates be located <i>outside</i> the <i>jupyter_installation_location</i> directory. As a result, the certificate files won't need to be moved after subsequent installations and the <code>config.conf</code> file can continue to point to the certificates.</p> <p>For example:</p> <pre>CERTIFICATES_DIRECTORY_PATH=// myjupyterserver.mycompany.com/ certificates</pre>
PROXY_CERTIFICATE_FILE_PATH	<p>If using SSL, enter the path, in Privacy Enhanced Mail (PEM) format, to the certificate file for the Jupyter Server.</p> <p>For example:</p> <pre>PROXY_CERTIFICATE_FILE_PATH=// myjupyterserver.mycompany.com/ certificates/ myjupyterserver.chained.pem</pre>
PROXY_KEY_FILE_PATH	<p>If using SSL, enter the path, in Privacy Enhanced Mail (PEM) format, to the certificate private key file for the Jupyter Server.</p> <p>For example:</p> <pre>PROXY_KEY_FILE_PATH=// myjupyterserver.mycompany.com/ certificates/myjupyterserver.my company.com.rsa.key</pre>
DOCKER_IMAGES_PATH=../../images	<p>If the location of your Docker images changes, update the path to the new location.</p>
HOST_NAME=\$(hostname)	<p>You should not need to edit the hostname value. It is resolved automatically if the hostname is set correctly. To check this, type <code>hostname</code> on a command line. The fully qualified name of the computer should be returned. If it is incorrect, you can edit the <code>HOST_NAME</code> value and add the fully qualified name.</p> <p>For example, type the following:</p> <pre>HOST_NAME=myjupyterserver.mycompany .com</pre>
HOST_PORT=8000	<p>The port number of the Jupyter Notebook hub.</p>

Parameter	Description
COGNOS_HOST	<p>COGNOS_HOST is an optional parameter that points the Jupyter server to the Cognos Analytics host. By default, the Jupyter server uses the Dispatcher URI for External Applications environment parameter from Cognos Configuration. However, if required, it can be overwritten here.</p> <p>Valid examples: <code>https://cognos.domain.com:9300</code>, <code>http://9.23.132.233:9300</code>, or <code>http://another-cognos-host.com</code></p> <p>Note: localhost or 127.0.0.1 cannot be used.</p>
SERVER_LIMIT=0	Specifies the maximum number of users that can be connected at one time. When set to 0 (the default value), no limit is enforced.
MEM_LIMIT=	<p>Specifies the memory limit for each user's container.</p> <p>The value can either be an integer (bytes) or a string with a K, M, G or T prefix.</p> <p>Examples:</p> <p><code>MEM_LIMIT=150M</code></p> <p><code>MEM_LIMIT=2G</code></p> <p>When there is no value (the default), the user container is allocated the memory that it requires.</p>
CULL_TIMEOUT	Specifies the idle time of each container, after which the cull service will remove them (default is 3600 seconds).
LOG_INTERVAL	Specifies, for each container, the time interval between log entries in <code>jupyter_installation_location/dist/logs/timestamp_folder/performance.txt</code> (default is 300 seconds).

Important: After you make any changes to the `config.conf` file, you must follow these steps:

- i) Run the `dist/scripts/unix/build.sh` for Linux or the `dist/scripts/windows/build.bat` for Windows script to apply your changes.
- ii) Run the `dist/scripts/unix/startup.sh` for Linux or the `dist/scripts/windows/startup.bat` for Windows script to see your changes.

2. Update Cognos Configuration as follows:

- a) Start Cognos Configuration.
- b) Click **Environment** and set the value for the **Dispatcher URI for External Applications** property.

Tip: If you set the `COGNOS_HOST` parameter in the `config.conf` file, you do not need to also set the **Dispatcher URI for External Applications** property in Cognos Configuration.

c) Set the value for the **Gateway URI** property.

Tip: You must set this value even if you have not explicitly configured an IIS or Apache gateway.

Examples

If no IIS or Apache gateway is configured in front of Cognos Analytics, the **Gateway URI** value is typically the URL to the Cognos server, for example, `https://cognos-host:9300/bi/v1/dispatch`

If another gateway is in place, the **Gateway URI** value must match it, for example, `https://gateway-host:443/ibmcognos/bi/v1/dispatch`

Configuring the Cognos Analytics gateway for Jupyter Notebook Server

If you installed the Cognos Analytics gateway and you want to integrate Cognos Analytics for Jupyter Notebook Server, you must edit the existing gateway configuration.

About this task

WebSocket communication is used between the Jupyter Notebook Server and the browser client.

The proxy layer in the Cognos Analytics service manages the dispatching of `http://` and `https://` traffic to the Jupyter server. However, it cannot broker WebSocket requests. Therefore, Notebook WebSocket requests must bypass the Cognos Analytics services layer and connect directly with the Jupyter service behind the Cognos Analytics service.

When Cognos Analytics is using a gateway, you can configure the bypassing of WebSocket traffic by adding a Rewrite Rule to the proxy specification. For more information, see "Configuring the gateway" in the *IBM Cognos Analytics Installation and Configuration Guide*.

Procedure

1. If you are using an Apache gateway, complete step 5 in the section "Configuring Apache HTTP Server or IBM HTTP Server with Cognos Analytics" in the *IBM Cognos Analytics Installation and Configuration Guide*.
2. If you are using an IIS gateway, proceed as follows:
 - a) Install WebSocket Protocol support in IIS. For more information, see [WebSocket <websocket>](https://docs.microsoft.com/en-us/iis/configuration/system.webserver/websocket) ([https://docs.microsoft.com/en-us/iis/configuration/system.webserver/web socket](https://docs.microsoft.com/en-us/iis/configuration/system.webserver/websocket)).
 - b) Complete step 6d in the topic "Configuring IIS in Cognos Analytics".
3. Restart the Cognos Analytics service.

Notebook performance logs

Jupyter Notebook performance logs contain information such as memory usage and CPU usage for each Docker container. This is useful if you want to view the usage of notebooks or notebook schedules, as each one uses a separate Docker container.

The performance logs are created in `jupyter_installation_location/dist/logs/timestamp_folder/performance.txt`

Here is an example of log entries in the file `performance.txt`:

```
[2023-03-30 20:51:44]
[1] Container name: ca_jupyter_hub, CPU Usage: 0.00%, Memory Usage: 272.84MiB
[2] Container name: ca_jupyter_viewer, CPU Usage: 0.00%, Memory Usage: 79.75MiB
Total CPU Usage: 0.00%, Total Memory Usage: 352.60MiB

[2023-03-30 20:56:44]
[1] Container name: ca_jupyter_hub, CPU Usage: 0.00%, Memory Usage: 272.87MiB
[2] Container name: ca_jupyter_viewer, CPU Usage: 0.00%, Memory Usage: 79.75MiB
Total CPU Usage: 0.00%, Total Memory Usage: 352.62MiB

[2023-03-30 21:01:44]
[1] Container name: ca_jupyter_hub, CPU Usage: 0.00%, Memory Usage: 272.91MiB
[2] Container name: ca_jupyter_viewer, CPU Usage: 0.00%, Memory Usage: 79.75MiB
Total CPU Usage: 0.00%, Total Memory Usage: 352.66MiB
```

Performance logs are always enabled. By default, the logs are gathered every 5 minutes. You can change this time interval in the `config.conf` file by updating the variable `LOG_INTERVAL`.

The logs capture memory and CPU usage of the Docker containers that the server is running in.

A summary line is added to the bottom of every log that shows the total memory and CPU usage of all active containers in the Jupyter server. This is important to know, because every notebook and schedule creates a new Docker container with its own memory and CPU footprint.

Troubleshooting

If the performance logs are not always created at equal intervals, there might be excessive load on the system that is impacting the logging service. Read the performance logs to see if memory or CPU usage may be nearing its capacity.

If you change the `LOG_INTERVAL` value in the `config.conf` file, but the new interval isn't applied to the logs, you may have forgotten to re-run the `install.sh` script to apply the changes. Re-running `stop.sh` and `startup.sh` will not apply your change.

Securing Jupyter Notebook Server

You can secure your Jupyter Notebook Server installation with SSL encryption using SSL certificates.

Note: If the Cognos Analytics server is secured with SSL, then the Jupyter Notebook server must also be secured with SSL. Similarly, if the Cognos Analytics server is **not** secured with SSL, the Jupyter Notebook server must also **not** be secured with SSL.

About this task

For a demonstration of how to secure Jupyter Notebook Server, [watch this video](#).

Procedure

1. Update the `config.conf` file for SSL encryption.
 - a) Set the value for `CERTIFICATES_DIRECTORY_PATH` with the path to the directory containing the authority certificates for the Jupyter server.
 - b) Set the value for `PROXY_CERTIFICATE_FILE_PATH` with the path to the certificate file for the Jupyter server.
 - c) Set the value for `PROXY_KEY_FILE_PATH` with the path to the private key file for the Jupyter server.

Tip: For more information see “Configuring Jupyter Notebook Server” on page 8.

2. Ensure that the administrator specifies `https`, rather than `http`, when they enable IBM Cognos Analytics for Jupyter Notebook. For more information, see *Managing IBM Cognos Analytics*.

3. Register the Jupyter server with the Cognos Analytics server as a trusted third party host.

Regardless if Cognos Analytics server is set up for SSL, you must still register the Jupyter server in the Cognos Analytics trusted service store. Cognos Analytics will not forward a request to an https target without first verifying (by certificate) that the target is trusted and genuine.

This involves importing a copy of the certificate for the secured Jupyter server to the Cognos Analytics trusted service store using the `ThirdPartyCertificateTool` utility provided with Cognos Analytics, in the `installation_location/bin` directory. For more information, see "ThirdPartyCertificateTool commands and examples" in the *IBM Cognos Analytics Installation and Configuration Guide*.

For example, to import a certificate, type the following on a command line at the computer where Cognos Analytics is installed:

```
ThirdPartyCertificateTool -i -T -p NoPassWordSet -r  
fully_qualified_pathname_of_jupyter_certificate_file_in_pem_format
```

4. **Only** if the Cognos Analytics server is also set up for SSL, register the Cognos Analytics server with the Jupyter server as a trusted third party host.
 - a) On the computer where Jupyter Server is installed, create a directory where the certificates will be stored.
 - b) Edit the file `config.conf` and set the `CERTIFICATES_DIRECTORY_PATH` parameter to point to the directory that you just created.
 - c) For each instance of Cognos Analytics that will connect to the Jupyter Server, copy the certificate in Privacy Enhanced Mail (PEM) format for the Cognos Analytics server into the certificates directory that you configured in step "4.b" on page 13.

Important: Even though the certificates must be in PEM format, they must have `.crt` file extensions.

- d) Rebuild the image:

In Linux, run `jupyter_installation_location/dist/scripts/unix/build.sh`

In Windows, run `jupyter_installation_location/dist/scripts/windows/build.bat`

- e) Restart the server:

In Linux, run `jupyter_installation_location/dist/scripts/unix/start.sh`

In Windows, run `jupyter_installation_location/dist/scripts/windows/startup.bat`

Results

The Jupyter Notebook Server is secured with SSL encryption.

Upgrading IBM Cognos Analytics for Jupyter Notebook Server

You can upgrade IBM Cognos Analytics for Jupyter Notebook Server to a newer version. Alternatively, you can update the Python packages in your existing installation of IBM Cognos Analytics for Jupyter Notebook Server.

Important: If the IBM Cognos Analytics for Jupyter Notebook server and its associated libraries are customized by users, support will be provided on a "best effort" basis only.

Upgrading your installation for Linux

You can install a newer version of IBM Cognos Analytics for Jupyter Notebook Server without manually uninstalling your current version.

Note: This task involves specifying a **different location** as your installation directory.

About this task

For a demonstration of how to upgrade your Jupyter Server installation, [watch this video](#).

Procedure

1. Follow steps “1” on page 3 to “4” on page 4 in “Installing Jupyter Notebook Server on Linux” on page 2, specifying a different installation location when prompted.
2. If you are upgrading from Cognos Analytics version 11.1.6 or later, copy the files `additional_pip_packages.txt` and `additional_conda_packages.txt` from `current_jupyter_installation_location/dist/scripts/` and paste them into the corresponding folder in your new location: `new_jupyter_installation_location/dist/scripts/`.
3. If you are upgrading from Cognos Analytics version 11.1.5 or earlier, copy the contents of the file `additional_packages.txt` from `current_jupyter_installation_location/dist/scripts/` and paste it into one or both of the following files, as required:
 - `new_jupyter_installation_location/dist/scripts/additional_pip_packages.txt`
 - `new_jupyter_installation_location/dist/scripts/additional_conda_packages.txt`
4. Copy the file `current_jupyter_installation_location/dist/scripts/unix/config.conf` and paste it into the corresponding folder in your new location: `new_jupyter_installation_location/dist/scripts/unix/`.
5. Navigate to the `new_jupyter_installation_location/dist/scripts/unix` directory.
6. Type `./install.sh`

Results

Your current installation is uninstalled. Then all of the Jupyter Server Docker images are loaded from the `new_jupyter_installation_location/dist/images` directory and the Docker containers are started.

Upgrading your installation for Microsoft Windows

You can install a newer version of IBM Cognos Analytics for Jupyter Notebook Server without manually uninstalling your current version.

Note: This task involves specifying a **different location** as your installation directory.

About this task

For a demonstration of how to upgrade your Jupyter Server installation, [watch this video](#).

Procedure

1. Follow steps 1 to 4 in [Installing Jupyter Notebook server on Microsoft Windows 10](#), specifying a different installation location when prompted.
2. If you are upgrading from Cognos Analytics version 11.1.6 or later, copy the files `additional_pip_packages.txt` and `additional_conda_packages.txt` from `current_jupyter_installation_location/dist/scripts/` and paste them into the corresponding folder in your new location: `new_jupyter_installation_location/dist/scripts/`.
3. If you are upgrading from Cognos Analytics version 11.1.5 or earlier, copy the contents of the file `additional_packages.txt` from `current_jupyter_installation_location/dist/scripts/` and paste it into one or both of the following files, as required:
 - `new_jupyter_installation_location/dist/scripts/additional_pip_packages.txt`

- `new_jupyter_installation_location/dist/scripts/additional_conda_packages.txt`
- 4. Copy the file `current_jupyter_installation_location/dist/scripts/windows/config.conf` and paste it into the corresponding folder in your new location: `new_jupyter_installation_location/dist/scripts/windows/`.
- 5. Open a command prompt window with administrator privileges.
- 6. Navigate to the `new_jupyter_installation_location/dist/scripts/windows` directory.
- 7. Type `./install.bat`

Results

Your current installation is uninstalled. Then all of the Jupyter Server Docker images are loaded from the `new_jupyter_installation_location/dist/images` directory and the Docker containers are started.

Upgrading Python packages and R packages

You can add other Python packages or R packages. You can also update the versions of existing Python packages and R packages in your IBM Cognos Analytics for Jupyter Notebook Server installation.

To accomplish this task, you edit the files `additional_pip_packages.txt` and `additional_conda_packages.txt`.

Note: The files `additional_pip_packages.txt` and `additional_conda_packages.txt` follow the standard `requirements.txt` file format that is used in Python, as specified in the [PyPA Reference Guide](https://pip.pypa.io/en/stable/reference/pip_install/#requirements-file-format) (https://pip.pypa.io/en/stable/reference/pip_install/#requirements-file-format).

Before you begin

Decide whether you need to upgrade. Check to see which versions of Python packages are in your current installation.

Tip: To see what is in the modules available in the Notebook environment, type the following in a notebook cell:

```
`!pip list --isolated`
```

You can also, for most Python packages, load a module at runtime for the specific Notebook with the `pip` command. For example, type the following in a Notebook cell:

```
!pip install --user prettyplotlib`
```

About this task

For a demonstration of how to upgrade Python packages, [watch this video](#).

Procedure

1. Edit the file `jupyter_installation_location/dist/scripts/additional_pip_packages.txt`

Tip: You can add Python packages by specifying them in the `additional_pip_packages.txt` file.

2. Stop the server:

For Linux, run `jupyter_installation_location/dist/scripts/unix/stop.sh`

For Windows, run `jupyter_installation_location/dist/scripts/windows/stop.bat`

3. Rebuild the image:

For Linux, run `jupyter_installation_location/dist/scripts/unix/build.sh`

For Windows, run `jupyter_installation_location/dist/scripts/windows/build.bat`

4. Restart the server:

For Linux, run `jupyter_installation_location/dist/scripts/unix/start.sh`

For Windows, run `jupyter_installation_location/dist/scripts/windows/startup.bat`

5. Edit the file `jupyter_installation_location/dist/scripts/additional_conda_packages.txt` and repeat steps 2-4.

Tip: You can add both Python packages and R packages by specifying them in the `additional_conda_packages.txt` file.

Adding additional Ubuntu operating system packages

You can add operating system packages into your Jupyter notebook server.

To do this, edit the `install_dir>/dist/scripts/additional_os_packages.txt`.

Note: Each package name must be written on a new line.

Procedure

1. Locate the `install_dir>/dist/scripts/additional_os_packages.txt`.
2. Add desired package on a new line.
3. Save the file.
4. Run the server `install.sh` script for Linux or the `install.bat` script for Microsoft Windows.

Troubleshooting IBM Cognos Analytics for Jupyter Notebook Server

To troubleshoot issues with Jupyter Notebook Server, you can use this Docker command: `docker logs container_id`

JupyterHub can provide useful information to help you diagnose a problem. For IBM Cognos Analytics for Jupyter Notebook Server, type the following:

```
docker logs ca_jupyter_hub
```

For more information, see [Troubleshooting](https://jupyterhub.readthedocs.io/en/latest/troubleshooting.html) (<https://jupyterhub.readthedocs.io/en/latest/troubleshooting.html>) on the JupyterHub web site.

No space left on device when installing Jupyter Server error message

Refer to official [Docker documentation](#) on how to free up space.

Note:

Docker stores all of its content (containers, images, and volumes) within `/var/lib/docker` by default.

The default storage location can be changed by adding the "data-root" key to the `daemon.json` file.

More information can be found on the docker daemon configuration file here:

<https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file>

<https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file>

Chapter 2. Enabling IBM Cognos Analytics for Jupyter Notebook

Administrators can configure IBM Cognos Analytics to connect to a computer that is running IBM Cognos Analytics for Jupyter Notebook.

Before you begin

IBM Cognos Analytics for Jupyter Notebook must be installed on another computer. For more information, see "Installing IBM Cognos Analytics for Jupyter Notebook" in *Installing and configuring Cognos Analytics*.

About this task

For a demonstration of how to enable IBM Cognos Analytics for Jupyter Notebook, [watch this video](#).

Procedure

1. Note the computer name where IBM Cognos Analytics for Jupyter Notebook is installed.
2. Go to **Manage > Configuration > System**, and select **Environment**.
3. In the **Jupyter service location** field, enter the following URL:

`http://Jupyter_Notebook_server_name:port_number`

Tip: Use `https://` if you have configured SSL on the Jupyter Notebook server. Note that if the Cognos Analytics server is secured with SSL, then the Jupyter Notebook server must also be secured with SSL.

4. Click **Apply**.

Results

The configuration change is saved and propagated to all dispatchers. You do not need to restart the service for users to connect to Jupyter Notebook.

What to do next

Ensure that you have assigned either the Notebook capability or roles that include the Notebook capability to your intended Jupyter Notebook users. For more information, see "Notebook capability" in the *IBM Cognos Analytics Managing Guide*. After you complete this task, users can start working with Jupyter Notebook in IBM Cognos Analytics.

Chapter 3. Getting started with Notebook

If IBM Cognos Analytics for Jupyter Notebook is enabled in IBM Cognos Analytics, you can work with notebook documents.

You can work with notebook documents, also referred to as notebooks, much like other content in Cognos Analytics.

Note: Before working with notebooks in Cognos Analytics, you should know how to work with and develop notebooks in Jupyter Notebook. You should be familiar with the Jupyter Notebook Editor.

CognosIBM Analytics for Jupyter Notebook supports the following notebook functionality:

Create and upload notebooks

Create notebooks. Edit, save, copy, and move notebooks. When you open a notebook, you work in the Jupyter Notebook Editor.

Create a notebook from the **Open menu** , by selecting the **New** option.

Upload notebooks that were created outside of Cognos Analytics. For more information, see [“Uploading external notebooks” on page 38](#).

Run and work with data in a notebook

The CADATAConnector API provides the `read_data()` and `write_data()` methods to read Cognos Analytics data sources and write to a data source. For example, you can run a notebook that reads external data, produces output, and saves the output as a data source.

You can use the Python and R programming languages in your notebooks.

For more information, see [“Reading data from a data source” on page 21](#) and [“Writing data to a data source” on page 25](#).

Include notebook output in a dashboard, story, or report

Embed the output from a notebook code cell in a dashboard, story, or report. For example, you have a notebook that creates a visualization that is not available in Cognos Analytics. You can add this visualization to a dashboard or story by embedding the code cell that creates it.

For more information adding notebook output to a dashboard or story, see *Adding a Notebook widget* in the *Dashboards and Stories User Guide*.

For more information adding notebook output to a report, see *Including output from a notebook* in the *Reporting User Guide*.

To ensure that the notebook visualizations display properly in dashboards, use some [coding best practices](#).

Samples


The samples show you how to work with notebooks in Cognos Analytics. If the samples are installed, you'll see them in **Team content > Samples > Notebooks**. For more information, see *Importing and configuring the Jupyter samples* in the *Samples Guide*.

Note: Before you can work with notebooks, CognosIBM Analytics for Jupyter Notebook must be installed and configured in your Cognos Analytics environment. Contact your administrator to confirm that you have the capability to work with notebooks. If you don't have the notebook capability, you won't see any of the functionality described in the following sections.

Creating a notebook


When you create a notebook, it opens in the Jupyter Notebook Editor.

Procedure

1. From the **Open menu** , click **New**, and then click **Notebook**.


Tip: If you don't see the **Notebook** selection, the notebook capability might not be enabled for you. Contact your administrator to find out.

The notebook opens in the Jupyter Notebook Editor.

2. To save the notebook, do the following steps:
 - a) Click **Save** , and then click **Save as**.
 - b) Choose the destination, type a name for the notebook, and click **Save**.

Notebook actions


After you've saved a notebook, you can perform actions on it from the welcome page or from **My content** or **Team content**.

Click the **Action menu**  for the notebook, and select one of the following actions:

Edit

Open the notebook in the Jupyter Notebook Editor.

Run

Run the notebook in the background. A message is displayed when the notebook starts running and when it finishes. Also a notification is added to your unread notifications .

View

When you view a notebook, you can see its contents but you can't edit or run it.

Properties

View the notebook properties. This is where you can schedule a notebook to run at regular intervals or on a specific date and time. For more information, see "Scheduling an entry" in the *IBM Cognos Analytics Getting Started Guide*.

Create a new job

Add a notebook to a job that runs the notebook at a scheduled time. For more information, see "Using jobs to schedule multiple entries" in the *IBM Cognos Analytics Getting Started Guide*.

Take ownership

Take ownership of the notebook. When you own a notebook, you can change the permissions for it. This action is available only in **Team content** since you already own the notebooks in **My content**. For more information, see "Simple and granular access permissions" in the *IBM Cognos Analytics Getting Started Guide*.

Copy or move

Copy or move the notebook. For more information, see *Copying or moving entries* in the *IBM Cognos Analytics Getting Started Guide*.

Create a shortcut

Create a shortcut to your notebook in **My content** or **Team content**.

Share

You can copy the URL of the notebook and use it in other places. If Cognos Analytics is connected to a collaboration tool, such as Slack, you can send a link to your notebook to other users. For more information, see "Sharing content" in the *IBM Cognos Analytics Getting Started Guide*.

Remove from Recent

Remove the notebook from the **Recent** view. It still exists in **My content** or **Team content**.

Delete

Delete the notebook from Cognos Analytics.

Tip: You can also access these actions in **My content** or **Team content** from the actions toolbar for a notebook. A subset of these actions is available if you do not have the **Edit notebook** capability.

Reading data from a data source

You can read Cognos Analytics data in a notebook using the Python or R programming languages.

You can read the following types of Cognos Analytics data sources in a notebook:

- Uploaded CSV or XLS files
- Data sets
- Data modules
- Framework Manager packages, including OLAP data

Note: Cognos Analytics does not support reading from data that requires user input. For example, a query subject in Framework Manager with a parameterized filter.

To quickly insert the `read_data()` method in a notebook cell, do the following steps:

1. Create or edit a notebook and position your cursor in the cell after which you want to do the read.

2. Click **Sources** .

3. Navigate to a data source, select it, and then click **Open**. The method is inserted in the cell with the data source to read specified.

Alternatively, you can type the code in a cell:

In Python

```
data = CADataConnector.read_data(parameters as described in the following sections)
```

In R

```
data <- CADataConnector$read_data(parameters as described in the following sections)
```

One of the following items is returned from the method:

DataFrame

Returned by default.

List of DataFrame

Returned when the **sheet_name** parameter is specified and multiple sheets are requested.

Iterator of DataFrame

Returned when the **chunksize** or **iterator** parameter is specified.

Parameters common to all data sources

For examples of how to code the `read_data()` method, see [“Python notebook examples” on page 29](#).

The following parameters can be specified for all supported data source types:

Parameter	Required or optional	Description
path id	Specify one of path or id	<p>Path to the data source. If the data source is in My content, specify <code>.my_folders</code> at the start of the path. If the data source is in Team content, specify <code>.public_folders</code> at the start of the path. For example, to specify a file called <code>sales-notebook</code> that is stored in My content, specify</p> <pre>path=".my_folders/sales-notebook"</pre> <p>The ID of the file. For information about how to get the ID of a file, see “Finding the ID of a file” on page 28. For example:</p> <pre>id="i1F8D76C0FAD34J9CA50118746935D9X7"</pre>
usecols	optional	<p>List of integers or strings that identifies a subset of columns to be returned. You can specify one or more columns by the position number of the column in the file or by column name. For example, to request columns City and Quantity, specify</p> <p>In Python</p> <pre>usecols=["City", "Quantity"]</pre> <p>In R</p> <pre>usecols=list("City", "Quantity")</pre> <p>If not specified, then all columns are returned.</p> <p>Note: This parameter is not applicable for OLAP data in a Framework Manager package.</p>
chunksize	optional	<p>An integer. If specified, the method returns an iterator. The value specifies the number of rows to return each time the method is invoked. For example, to return 3 rows, specify</p> <pre>chunksize=3</pre> <p>If both chunksize and iterator are not specified, then a DataFrame is returned.</p>

Parameter	Required or optional	Description
iterator	optional	<p>If specified, the method returns an iterator. Specifying</p> <pre>iterator=True</pre> <p>without chunksize specified returns one row each time the method is invoked.</p> <p>Specifying <code>iterator=True</code> with chunksize specified, returns chunksize rows each time the method is invoked.</p> <p>Reset the iterator using the reset method:</p> <p>In Python</p> <pre>iterator.reset()</pre> <p>In R</p> <pre>iterator\$reset()</pre> <p>method.</p>
nrows	optional	<p>The maximum number of rows returned. Specify an integer.</p> <p>If omitted, the maximum number of rows returned is 10,000. If</p> <pre>nrows=0</pre> <p>is specified, then the column headings are returned.</p>

XLS file parameters

Uploaded files are treated as sheets of data. Reading one of these data sources returns the rows for the specified sheet of data in the data source.

The following parameter applies only to XLS files:

sheet_name

Optional. Integer, string, or a list of integers or strings. Specifies the integer position of a specific sheet or tab in an XLS file. The first sheet is 0. If you specify a list of strings or integers, then a list of the corresponding sheets is returned. If not specified, the value defaults to 0. Example:

In Python

```
sheet_name=["sheet2", "sheet3"]
```

In R

```
sheet_name=list("sheet2", "sheet3")
```

Data module parameters

Reading data from a data module must reflect the modeling done on the data source. For example, a join between two tables and aggregation types to apply.

The following parameters apply only to data modules:

Parameter	Required or optional	Description
table_name	Optional	<p>String or list of strings. Restrict the scope of the request to specific table(s) in the data module. Specify the table(s) by table name(s).</p> <p>If you omit table_name, the <code>read_data()</code> method returns a DataFrame that contains the names of the tables defined in the data module. Example:</p> <p>In Python</p> <pre>table_name=["table1","table2"]</pre> <p>In R</p> <pre>table_name=list("table1","table2")</pre>
calculation	Optional	String or list of strings. Read a calculation from a data module. Specify a calculated column in the data module.

Package parameters

Reading data from a package must reflect the modeling done on the data source. For example, the relationship between objects in the package.

The following parameters apply only to packages:

Parameter	Required or optional	Description
query_subject	Optional	<p>String or list of strings. Restrict the scope of the request to the specific query subjects(s) in the package. Specify the query subject(s) by query subject name(s). Example:</p> <pre>query_subject="query1"</pre>
folder_name	Optional	<p>String or list of strings. Restrict the scope of the request to a folder in the package. Displays the contents of all folder(s) with that name in the package. If you want to see only the content of folder c that's in folder b that's in folder a specify the folder_name parameter as a list:</p> <p>In Python</p> <pre>folder_name=["a","b","c"]</pre> <p>In R</p> <pre>folder_name=list("a","b","c")</pre> <p>This returns only the folder c in this path, even if there are other folders with the name c in the package.</p>
metadata	Optional. Applies only to OLAP data in a Framework Manager package.	<p>Value can be True or False.</p> <p>Specifying</p> <pre>metadata=True</pre> <p>returns the query subjects in the package. Specifying</p>

Parameter	Required or optional	Description
		<pre>metadata=False</pre> <p>returns the data in the package. If not specified, then a default value of False is used.</p>

Note: You can use either single or double quotation marks in a method but not a mix of both.

Writing data to a data source

You can write Cognos Analytics data in a notebook using the Python or R programming languages.

You can save or create the data source in **My content** or **Team content**.

To write data to the data source, specify the following code in a notebook cell:

In Python

```
data = CADATAConnector.write_data(parameters as described in the following sections)
```

In R

```
data <- CADATAConnector$write_data(parameters as described in the following sections)
```

Parameter	Required or optional	Description
data	required	Pandas.DataFrame. Contains the table of data, which is written to the file.
path id	Specify one of path or id	<p>Path to the data source. If the data source is in My content, specify <code>.my_folders</code> at the start of the path. If the data source is in Team content, specify <code>.public_folders</code> at the start of the path. For example, to specify a file called <code>sales-notebook</code> that is stored in My content, specify</p> <pre>path=".my_folders/sales-notebook"</pre> <p>The ID of the file. For information about how to get the ID of a file, see “Finding the ID of a file” on page 28. For example:</p> <pre>id="i1F8D76C0FAD34J9CA50118746935D9X7"</pre>

Parameter	Required or optional	Description
mode	optional	<p>x Create the file and write to it. If the file exists, the method fails with an exception. Default value.</p> <p>w Overwrite any data that exists in the file. The columns that you are writing to the file must match the ones in the file. Create the file if it doesn't exist.</p> <p>a Append to the end of the file. The columns that you are adding to the file must match the ones in the file. Create the file if it doesn't exist.</p> <p>Example:</p> <pre>mode="w"</pre>

Note: You can use either single or double quotation marks in a method but not a mix of both.

For examples of how to code the `write_data()` method, see [“Python notebook examples” on page 29](#) and [“R notebook examples” on page 32](#).

Searching for data objects

You can search for data objects in a notebook using the Python or R programming languages.

The `search_data()` method is used to find data objects so that they can be further processed using the `CADDataConnector`.

The result provides a list of data objects and their connection paths, which may be copy/pasted for use with the various connection and access functions in the `CADDataConnector`.

To search for data objects, specify the following code in a notebook cell:

In Python

```
data = CADDataConnector.search_data(parameters as described in the following sections)
```

In R

```
data <- CADDataConnector::search_data(parameters as described in the following sections)
```

Parameter	Required or optional	Description
query='search_term'	optional	<p><code>search_term</code> can be any data object.</p> <p>If no parameters are supplied, that is, <code>CADDataConnector.search_data()</code>, all data objects available are selected, to a maximum of 50. See an example of the resulting output.</p>

Parameter	Required or optional	Description
<code>types=types</code>	optional	<p><code>types</code> can be replaced with any number of (separated by either ' ' or ';') the following data object types:</p> <ul style="list-style-type: none"> • uploadedFile • dataset • dataset2 • module • package <p>Example:</p> <pre>types='module,dataset'</pre> <p>If <code>types=types</code> is not specified, the default is all of the types listed above.</p>
<code>max=number</code>	optional	<p><code>number</code> is any number greater than zero.</p> <p>Example:</p> <pre>max=20</pre> <p>If <code>max=number</code> is not specified, the default is</p> <pre>max=50</pre> <p>Use a higher number if you see a message that the maximum results were returned.</p>

Note: You can use either single or double quotation marks in a method but not a mix of both.

Example output when no parameters are specified

When no parameters are supplied, the output appears in a list:

```
Results found: 50. ( see more results by using "max=" parameter with a number higher than 50 )

Type          Open path
package       .public_folders/Samples/Models/GO sales (analysis)
package       .public_folders/Samples_LG_DQ/Models/GO Data Warehouse (query)
package       .public_folders/Samples/Models/GO sales (query)
package       .public_folders/Samples_LG_DQ/Models/GO Sales (query)
package       .public_folders/Samples/Models/GO data warehouse (query)
package       .public_folders/Samples_LG_DQ/Models/GO Data Warehouse (analysis)
package       .public_folders/Samples/By feature/Audit
package       .public_folders/Samples/Models/GO data warehouse (analysis)
package       .public_folders/Samples_LG_DQ/Models/GO Sales (analysis)
package       .public_folders/Samples/Data/Sporting goods company
uploadedFile  .public_folders/Samples/By feature/Notebooks/Data/Source files/Notebook data/
Weather
uploadedFile  .public_folders/Samples/By feature/Notebooks/Data/Source files/
Hospital_floor_plan.xlsx
.....
```

Python notebook examples

```
CADDataConnector.search_data('boston')
```

Returns everything with 'boston' in the name or data, to a maximum of the default number of results.

```
CADDataConnector.search_data(types='module,package',max=100)
```

Returns only modules or packages to a maximum of 100 results.

```
CADDataConnector.search_data(max=20)
```

Returns everything to a maximum of 20 results.

Note: All or none of the parameters may be used. If the query is to be used, it must be the first parameter. The other parameters are named and are not position-dependent.

R notebook examples

```
CADDataConnector::search_data('boston')
```

Returns everything with 'boston' in the name or data, to a maximum of the default number of results.

```
CADDataConnector::search_data(types='module,package',max=100)
```

Returns only modules or packages to a maximum of 100 results.

```
CADDataConnector::search_data(max=20)
```

Returns everything to a maximum of 20 results.

Note: All or none of the parameters may be used. If the query is to be used, it must be the first parameter. The other parameters are named and are not position-dependent.

File paths that contain a forward slash

To reference a file or folder name that contains a folder slash in the CADDataConnector API, you must use a specific syntax.

To enable the CADDataConnector API to differentiate between a forward slash that indicates folder structure and a forward slash that is part of a file or folder name, enclose each folder or file name in single or double quotation marks, separate each folder or file name in the path with a comma, and enclose the whole thing in square brackets.


In the following example, the path to a file named `my/data.csv` is `my/folder-1/my/folder-2/my/data.csv`.

```
data = CADDataConnector.read_data(path=["my/folder-1","my/folder-2","my/data.csv"])
```

Finding the ID of a file

To reference a file in the CADDataConnector API, you can use its ID.

Procedure

1. In a content page, such as **My Content** or **Team content**, click the **Action menu**  for a file, and select **Properties**.
2. On the **General** tab, click **Advanced**. The **ID** field contains the file ID.

Python notebook examples

Here are some examples that demonstrate how to work with Cognos Analytics data sources in a notebook using the Python programming language.

Tip: To switch your notebook's programming mode from **R** to **Python**, from the **Kernel** menu, select **Change kernel > Python**.

Reading a file

The following example reads a file called `SampleFile_GOSales.xls`, outputs the entire file, and then specifies the **nrows** and **usecols** parameters to output the **City** and **Quantity** columns for the first 2 rows of data in the file.

```
In [1]: data = CDataConnector.read_data(path=".public_folders/jmdatasets/SampleFile_GOSales.xls")

In [2]: data.head()
Out[2]:
```

	Retailer country	Province or State	City	Postal code	Short postal code	Order method type	Retailer type	Retailer	Product line	Product type	Year	Quarter	Quantity	Unit cost	Unit price	Unit sale price	Re
0	United Kingdom	West Midlands	Birmingham	B29 5DW	B2	Web	Outdoors Shop	Seamate Chandeliers	Camping Equipment	Cooking Gear	2014	Q3	1868	2.90	6.59	6.26	11
1	Canada	Saskatchewan	Regina	S1J 3C5	S1J	Web	Sports Store	Ultra Sports	Camping Equipment	Cooking Gear	2017	Q2	3659	3.01	6.59	6.26	22
2	United States	Florida	Miami	33176	33176	Web	Sports Store	Island Sports	Camping Equipment	Cooking Gear	2016	Q2	1218	2.93	6.59	6.48	7
3	United Kingdom	West Midlands	Birmingham	B20 3BA	B2	Web	Department Store	Leisure Land	Camping Equipment	Cooking Gear	2015	Q4	1638	2.93	6.59	6.19	10
4	United Kingdom	West Midlands	Birmingham	B20 3BA	B2	Web	Department Store	Leisure Land	Camping Equipment	Cooking Gear	2016	Q4	1315	2.93	6.59	6.26	8

5 rows x 22 columns

```
In [ ]:

In [5]: data = CDataConnector.read_data(path=".public_folders/jmdatasets/SampleFile_GOSales.xls",
data
nrows=2, usecols=['City', 'Quantity'])
Out[5]:
```

	City	Quantity
0	Birmingham	1868
1	Regina	3659

```
In [ ]:
```

Reading part of a file

Specify **iterator** and **chunksize** to work with chunks of data, rather than the whole data source all at once. The following example reads 20,000 rows at a time from the `SampleFile_GOSales.xls` file.

```
In [23]: iter = CDataConnector.read_data(path=".public_folders/jmdatasets/SampleFile_GOSales.xls",
chunksize = 20000)

In [24]: for chunk in iter:
display(chunk.head(2))
```

	Retailer country	Province or State	City	Postal code	Short postal code	Order method type	Retailer type	Retailer	Product line	Product type	Year	Quarter	Quantity	Unit cost	Unit price	Unit sale price	Reven
0	United Kingdom	West Midlands	Birmingham	B29 5DW	B2	Web	Outdoors Shop	Seamate Chandeliers	Camping Equipment	Cooking Gear	2014	Q3	1868	2.90	6.59	6.26	1168
1	Canada	Saskatchewan	Regina	S1J 3C5	S1J	Web	Sports Store	Ultra Sports	Camping Equipment	Cooking Gear	2017	Q2	3659	3.01	6.59	6.26	2296

2 rows x 22 columns

	Retailer country	Province or State	City	Postal code	Short postal code	Order method type	Retailer type	Retailer	Product line	Product type	Year	Quarter	Quantity	Unit cost	Unit price	Unit sale price	Reven
0	Germany	Bayern	München	80800	80800	Web	Eyewear Store	Weltblick	Personal Accessories	Eyewear	2017	Q1	1082	25.70	61.84	59.636222	64158.
1	United Kingdom	Greater London	London	E14 7LB	E1	Web	Sports Store	Total Sports	Personal Accessories	Eyewear	2015	Q4	206	28.15	61.84	60.230000	12383.

Writing to a file

The following example writes the contents of a DataFrame table of data in to a file called `regions sales`, which is then stored in **My content**.

```
In [20]: import pandas as pd
newData = pd.DataFrame({'Regions': ['Americas', 'Europe', 'Asia'], 'Sales': [300, 400, 500]})

In [21]: newData
Out[21]:
```

	Regions	Sales
0	Americas	300
1	Europe	400
2	Asia	500

```
In [22]: CDataConnector.write_data(newData, path=".my_folders/region sales", mode="w")
```

Reading a data module

A data module has relationships, aggregations, calculated columns, and so on, defined on its data. The `read_data()` method defines a section of data from a data module by selecting columns from the tables

in the module. The data that is returned includes the relationships, aggregations, calculated columns, and so on, defined in the data module.

Specifying the `read_data()` method without the **table_name** parameter returns a **DataFrame** that contains the names of all the tables defined in the data module, as shown in the following example:

```
In [26]: data = CDataConnector.read_data(path=".public_folders/jmdatasets/sample_data_module")
data
Out[26]:
```

	Name	Table Name	Type
0	American Time Use Xlsx	M1_American_time_use_xlsx	Data
1	Banking Loss Events Xlsx	M1_Banking_loss_events_xlsx	Data

In the following example, the `M1_American_time_use_xlsx` table in the `sample_data_module` data module is read and the columns **Year** and **Children** are returned.

```
In [28]: data = CDataConnector.read_data(path=".public_folders/jmdatasets/sample_data_module",
data
table_name="M1_American_time_use_xlsx",
usecols=["Year", "Children"])
Out[28]:
```

	Year	Children
0	2004	0.844863
1	2009	0.863970
2	2010	0.881495
3	2006	0.959833
4	2008	0.902300
5	2012	0.832634
6	2003	0.872129
7	2005	0.938406
8	2007	0.914562
9	2011	0.835999

Reading a calculated column from a data module

Use the **calculation** parameter to read a calculated column from a data module.

The following example reads 2 calculations from the `calculation_data_module` data module.

```
In [5]: # Multiple calculations can be read at once
data = CDataConnector.read_data(
    path=".my_folders/calculation_data_module",
    calculation=['half_population', 'double_population']
)
data
Out[5]:
```

	half_population	double_population
0	16940919.0	67763676

Reading a package

Like a data module, a package has relationships, aggregations, calculated columns, and so on, defined on its data. In addition, a package logically groups data into query subjects and folders. You can use the `read_data()` method to navigate through the structure of a package by using the **query_subject** and **folder_name** parameters.

The following code reads the `Go_data_warehouse` package:

```
In [1]: toplevel = CDataConnector.read_data(path=".public_folders/jmdatasets/GO_data_warehouse (query)")
In [2]: toplevel
Out[2]:
```

	Name	Query Subject	Type
0	HR (query)	N/A	Folder
1	Sales and Marketing (query)	N/A	Folder
2	Finance (query)	N/A	Folder
3	Filters	N/A	Folder

Specifying the **folder_name** parameter returns the contents of all folder(s) with that name in the package. If you want to return only the content of one specific folder, for example folder `c`, that's inside folder `b`, that's inside folder `a`, put a list of the folder names in the **folder_name** parameter. The

following example uses the **folder_name** parameter to get all of the query subjects in the Employee expense folder:

```
In [3]: CADataConnector.read_data(path=".public_folders/jmdatasets/GO data warehouse (query)", folder_name=toplevel['Name'][0])
```

```
M In [4]: hr
```

```
Out[4]:
```

	Name	Query Subject	Type
0	Employee expense (query)	N/A	Folder
1	Employee expense plan (query)	N/A	Folder
2	Employee position summary (query)	N/A	Folder
3	Employee ranking (query)	N/A	Folder
4	Employee recruitment (query)	N/A	Folder
5	Employee succession (query)	N/A	Folder
6	Employee summary (query)	N/A	Folder
7	Employee survey (query)	N/A	Folder
8	Employee survey target (query)	N/A	Folder
9	Employee training (query)	N/A	Folder

```
In [5]: exp = CADataConnector.read_data(path=".public_folders/jmdatasets/GO data warehouse (query)", folder_name=hr['Name'][0])
```

```
In [6]: exp
```

```
Out[6]:
```

	Name	Query Subject	Type
0	Employee expense fact	[Employee expense (query)][Employee expense f...	Data Sheet
1	Account	[Employee expense (query)][Account]	Data Sheet
2	Employee by manager	[Employee expense (query)][Employee by manager]	Data Sheet
3	Employee by region	[Employee expense (query)][Employee by region]	Data Sheet
4	Employee expense	[Employee expense (query)][Employee expense]	Data Sheet
5	Organization	[Employee expense (query)][Organization]	Data Sheet
6	Organization (consolidated)	[Employee expense (query)][Organization (cons...	Data Sheet
7	Position-department	[Employee expense (query)][Position-department]	Data Sheet
8	Time	[Employee expense (query)][Time]	Data Sheet
9	YTD Time	[Employee expense (query)][YTD Time]	Data Sheet

The following code returns the query items that are in the Employee expense fact query subject:

```
In [20]: expfacts = CADataConnector.read_data(path=".public_folders/jmdatasets/GO data warehouse (query)",
query_subject=[exp['Query Subject'][0]])
```

```
In [21]: expfacts
```

```
Out[21]:
```

	Expense unit quantity	Expense total	Expense unit quantity YTD	Expense total YTD
0	4.004779e+05	168696729.2	4.004779e+05	168696729.2

The following code returns the query items that are in the Account query subject:

```
M In [24]: accounts = CADataConnector.read_data(path=".public_folders/jmdatasets/GO data warehouse (query)",
accounts.columns,
query_subject=[exp['Query Subject'][1]])
```

```
Out[24]: Index(['Account name (level 1)', 'Account name (level 2)',
'Account name (level 3)', 'Account name (level 4)',
'Account name (level 5)', 'Account name (level 6)',
'Account name (level 7)', 'Account name (level 8)',
'Account name (level 9)', 'Account name (level 10)',
'Account name (level 11)', 'Account name (level 12)',
'Account name (level 13)', 'Account name (level 14)',
'Account name (level 15)', 'Account name (level 16)', 'Account name',
'Account level', 'Account parent', 'Debit or credit',
'Account code (level 1)', 'Account code (level 2)',
'Account code (level 3)', 'Account code (level 4)',
'Account code (level 5)', 'Account code (level 6)',
'Account code (level 7)', 'Account code (level 8)',
'Account code (level 9)', 'Account code (level 10)',
'Account code (level 11)', 'Account code (level 12)',
'Account code (level 13)', 'Account code (level 14)',
'Account code (level 15)', 'Account code (level 16)', 'Account key',
'Account code', 'Account type code', 'Account class code'],
dtype='object')
```

The following code selects the first two query subjects in the Employee expense folder. **Expense total** is returned from the first query subject (**Employee expense fact**) and **Account code** is returned from the second query subject (**Account**).

```
M In [25]: exp_by_account = expfacts = CADataConnector.read_data(path=".public_folders/jmdatasets/GO data warehouse (query)",
query_subject=[exp['Query Subject'][0], exp['Query Subject'][1]],
usecols=['Expense total', 'Account code'])
```

```
In [26]: exp_by_account
```

```
Out[26]:
```

	Expense total	Account code
0	95723630.59	601100
1	39938062.73	601500
2	4008748.74	601600
3	10555741.81	605050
4	14514144.54	605500
5	3958400.79	605800

Reading the metadata for OLAP data in a Framework Manager package

The following code shows the **metadata** parameter set to true, which returns the query subjects in the package:

```
In [5]: data = CADATAConnector.read_data(
        path=".public_folders/PowerCubePackage",
        query_subject="[PowerCube].[Products].[Products]",
        metadata=True
    )
    display(data)
```

	Name	Query Subject
0	Products	[PowerCube].[Products].[Products].[Products]
1	Product line	[PowerCube].[Products].[Products].[Product line]
2	Product type	[PowerCube].[Products].[Products].[Product type]
3	Product	[PowerCube].[Products].[Products].[Product]

```
In [ ]:
```

R notebook examples

Here are some examples that demonstrate how to work with Cognos Analytics data sources in a notebook using the R programming language.

Tip: To switch your notebook's programming mode from **Python** to **R**, from the **Kernel** menu, select **Change kernel > R**.

A basic read

The following example reads a file called `SampleFile_GOSales.xls` and displays the first six rows of it.

```
In [3]: data <- CADATAConnector$read_data(path=".public_folders/jimdatasets/SampleFile_GOSales.xls")
        head(data)
```

Retailer country	Province or State	City	Postal code	Short postal code	Order method type	Retailer type	Retailer	Product line	Product type	...	Planned revenue
United Kingdom	West Midlands	Birmingham	B29 5DW	B2	Web	Outdoors Shop	Seamate Chandlers	Camping Equipment	Cooking Gear		12310.12
Canada	Saskatchewan	Regina	S1J 3C5	S1J	Web	Sports Store	Ultra Sports	Camping Equipment	Cooking Gear		24112.81
United States	Florida	Miami	33176	33176	Web	Sports Store	Island Sports	Camping Equipment	Cooking Gear		8026.62
United Kingdom	West Midlands	Birmingham	B20 3BA	B2	Web	Department Store	Leisure Land	Camping Equipment	Cooking Gear		10794.42
United Kingdom	West Midlands	Birmingham	B20 3BA	B2	Web	Department Store	Leisure Land	Camping Equipment	Cooking Gear		8665.85
United Kingdom	Greater London	London	E15 3JR	E1	Sales visit	Outdoors Shop	Jensen Mountaineering	Camping Equipment	Cooking Gear		19967.70

Reading a file

The following example reads a file called `SampleFile_GOSales.xls` and specifies the **nrows** and **usecols** parameters to output the **City** and **Quantity** columns for the first two rows of data in the file.

```
In [1]: data <- CADataConnector$read_data(
  path=".public_folders/jimdatasets/SampleFile_GOSales.xls",
  usecols=list('City', 'Quantity'),
  nrows=2
)
data
```

City	Quantity
Birmingham	1868
Regina	3659

Reading part of a file

Specify **iterator** and **chunksize** to work with chunks of data, rather than the whole data source all at once. The following example reads 20,000 rows at a time from the SampleFile_GOSales.xls file. As the file is read, the number of rows in the chunk is displayed.

```
In [8]: iter <- CADataConnector$read_data(
  path=".public_folders/jimdatasets/SampleFile_GOSales.xls",
  chunksize=20000
)
```

```
⌘ In [9]: chunk <- iter$get_chunk()
while(nrow(chunk) > 0) {
  print(nrow(chunk))
  chunk <- iter$get_chunk()
}
```

```
[1] 20000
[1] 20000
[1] 20000
[1] 5535
```

Writing to a file

The following example writes the contents of a DataFrame table of data in to a file called regions sales, which is then stored in **My content**.

```
In [1]: newData <- data.frame("Regions" = c("Americas", "Europe", "Asia"), "Sales" = c(300, 400, 500))
```

```
In [2]: newData
```

Regions	Sales
Americas	300
Europe	400
Asia	500

```
In [5]: CADataConnector$write_data(newData, path=".my_folders/region sales", mode="w")
```

Reading a data module

A data module has relationships, aggregations, calculated columns, and so on, defined on its data. The `read_data()` method defines a section of data from a data module by selecting columns from the tables in the module. The data that is returned includes the relationships, aggregations, calculated columns, and so on, defined in the data module.

Specifying the `read_data()` method without the **table_name** parameter returns a DataFrame that contains the names of all the tables defined in the data module, as shown in the following example:

```
In [6]: data <- CADataConnector$read_data(  
        path=".public_folders/Samples/Data/Coffee sales and marketing"  
        )  
data
```

Name	Table Name	Type
Sales Receipts	i201904_sales_reciepts	Data
Pastry Inventory	pastry_inventory	Data
Sales Targets	sales_targets	Data
Customer	customer	Data
Dates	Dates	Data
Product	product	Data
Sales Outlet	sales_outlet	Data
Staff	staff	Data
Generation	generations	Data
Spoilage_Expense	N/A	Calculation
beverages_sale	N/A	Calculation
beverages_sale_1	N/A	Calculation
beverages_sale_2	N/A	Calculation
beverages_sale_3	N/A	Calculation
beverages_sale_4	N/A	Calculation

In the following example, the `pastry_inventory` table in the `Coffee sales and marketing` data module is read and the columns **Date** and **Quantity Sold** are returned.


```
In [8]: data <- CADataConnector$read_data(
  path=".public_folders/Samples/Data/Coffee sales and marketing",
  table_name='pastry_inventory',
  usecols=list("Date", "Quantity Sold")
)
data|
```

Date	Quantity Sold
2019-04-04	48
2019-04-05	0
2019-04-11	127
2019-04-12	133
2019-04-13	123
2019-04-17	149
2019-04-19	159
2019-04-20	163
2019-04-21	161
2019-04-22	102
2019-04-23	93
2019-04-26	102
2019-04-01	46
2019-04-02	46
2019-04-03	41
2019-04-06	70
2019-04-07	104
2019-04-08	118
2019-04-09	91
2019-04-10	110
2019-04-14	167
2019-04-15	149
2019-04-16	144
2019-04-18	117
2019-04-24	105
2019-04-25	109
2019-04-27	77

Reading a package

Like a data module, a package has relationships, aggregations, calculated columns, and so on, defined on its data. In addition, a package logically groups data into query subjects and folders. You can use the `read_data()` method to navigate through the structure of a package by using the **query_subject** and **folder_name** parameters.

The following code reads the Go data warehouse package:

```
In [3]: toplevel <- CADataConnector$read_data(  
  path=".public_folders/Samples_DQ_10.2.2_DB2/Models/GO Data Warehouse (query)")  
  toplevel
```

Name	Query Subject	Type
HR (query)	N/A	Folder
Sales and Marketing (query)	N/A	Folder
Finance (query)	N/A	Folder
Filters	N/A	Folder

Specifying the **folder_name** parameter returns the contents of all folders with that name in the package. If you want to return only the content of one specific folder, for example folder c, that's inside folder b, that's inside folder a, put a list of the folder names in the **folder_name** parameter. The following example uses the **folder_name** parameter to get all of the query subjects in the Employee expense folder:


```
In [4]: hr <- CADataConnector$read_data(
  path=".public_folders/Samples_DQ_10.2.2_DB2/Models/GO Data Warehouse (query)",
  folder_name=toplevel[1,1]
)
hr
```

Name	Query Subject	Type
Employee expense (query)	N/A	Folder
Employee expense plan (query)	N/A	Folder
Employee position summary (query)	N/A	Folder
Employee ranking (query)	N/A	Folder
Employee recruitment (query)	N/A	Folder
Employee succession (query)	N/A	Folder
Employee summary (query)	N/A	Folder
Employee survey (query)	N/A	Folder
Employee survey target (query)	N/A	Folder
Employee training (query)	N/A	Folder

```
In [5]: exp <- CADataConnector$read_data(
  path=".public_folders/Samples_DQ_10.2.2_DB2/Models/GO Data Warehouse (query)",
  folder_name=hr[1,1]
)
exp
```

Name	Query Subject	Type
Employee expense fact	[Employee expense (query)].[Employee expense fact]	Data
Account	[Employee expense (query)].[Account]	Data
Employee by manager	[Employee expense (query)].[Employee by manager]	Data
Employee by region	[Employee expense (query)].[Employee by region]	Data
Employee expense	[Employee expense (query)].[Employee expense]	Data
Organization	[Employee expense (query)].[Organization]	Data
Organization (consolidated)	[Employee expense (query)].[Organization (consolidated)]	Data
Position-department	[Employee expense (query)].[Position-department]	Data
Time	[Employee expense (query)].[Time]	Data
YTD Time	[Employee expense (query)].[YTD Time]	Data

The following code returns the query items that are in the **Employee expense fact** query subject:

```
In [10]: expfacts <- CADataConnector$read_data(
  path=".public_folders/Samples_DQ_10.2.2_DB2/Models/GO Data Warehouse (query)",
  query_subject=list(exp[1,2])
)
expfacts
```

Expense unit quantity	Expense total	Expense unit quantity YTD	Expense total YTD
4004779	168696729	4004779	168696729

The following code returns the query items that are in the **Account** query subject:

```
In [14]: accounts <- CADataConnector$read_data(
  path=".public_folders/Samples_DQ_10.2.2_DB2/Models/GO Data Warehouse (query)",
  query_subject=list(exp[2,2])
)
accounts[0,]
```

Account name (level 1)	Account name (level 2)	Account name (level 3)	Account name (level 4)	Account name (level 5)	Account name (level 6)	Account name (level 7)	Account name (level 8)	na	Debit or credit
------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	----	-----------------

Reading the metadata for OLAP data in a Framework Manager package

The following code shows the **metadata** parameter set to true, which returns the query subjects in the package:

```
In [5]: data <- CADataConnector$read_data(
  path=".public_folders/powercube",
  query_subject="[powercube].[Products].[Products]",
  metadata=TRUE
)
data
```

Name	Query Subject
Products	[powercube].[Products].[Products].[Products]
Product line	[powercube].[Products].[Products].[Product line]
Product type	[powercube].[Products].[Products].[Product type]
Product	[powercube].[Products].[Products].[Product]

Uploading external notebooks

You can upload Jupyter Notebooks (.ipynb or .zip extension) that were created in a Jupyter environment outside of IBM Cognos Analytics.

The uploaded notebooks are matched against the following JSON schemas:

- <https://github.com/jupyter/nbformat/blob/master/nbformat/v3/nbformat.v3.schema.json> (version 3 (v3) notebooks)
- <https://github.com/jupyter/nbformat/blob/master/nbformat/v4/nbformat.v4.schema.json> (version 4 (v4) notebooks)

If a notebook doesn't conform to either of these schemas, the upload is rejected.


Jupyter Server creates all new notebooks using the v4 schema.

About this task

The process of uploading notebook files is similar to the process of uploading spreadsheets and text files that are used as data sources in Cognos Analytics. For example, the notebook files have the same size limitations as other uploaded file types. For more information, see [Upload data files](#).

Procedure

Use the following methods to upload the notebooks:

- From the **Open menu** , click **Upload data**. Locate the notebook files (.ipynb or .zip) on your local drive or on the LAN, and select one or multiple files to upload them.
- In the welcome page, drag one or multiple .ipynb or .zip files from your local drive onto the welcome page to activate the **Quick upload** functionality. When **Quick upload** appears, drop the files into the **Notebook** box.
- From **Team content** or **My content**, click **Upload data**. Locate the notebook files on your local drive or on the LAN, and select one or multiple files to upload them. The files are saved to the folder from which you initiated the upload.

Results

A notebook is created for each uploaded .ipynb file in **My content**. However, when the upload was initiated from a specific folder, the notebooks are created in that folder.

What to do next

You can seamlessly open an uploaded v4 notebook in the edit mode. To open a v3 notebook in the edit mode, the Jupyter server temporarily converts it to the v4 format. If you save the notebook, it is saved in the v4 format.

Tip: Older versions of Jupyter might not read the v4 format. To preserve the original notebook version, close it without saving.

You can view both v3 and v4 notebooks in the view-only mode. Because you can't interact with notebooks in this mode, there is no need to convert v3 notebooks to v4.

You can import uploaded v3 and v4 notebooks into a Cognos Analytics dashboard. The **Notebook** widget in the dashboards conforms to the v4 JSON schema so the visualizations in v4 notebooks display seamlessly in the dashboard. To ensure that the v3 notebook visualizations display properly in the dashboard, you must open and save the v3 notebooks in the edit mode before you import them into the dashboard.


Importing Watson Studio notebooks to Cognos Analytics

If you are using Cognos Analytics on Cloud, you may have been given access to a Watson Studio environment that is external to your Cognos Analytics environment. If so, you can import notebooks that were authored in the Watson Studio environment. You can then select a content cell from an imported notebook and add it to a report or dashboard.

Before you begin

Confirm that you have been given access to an external Watson Studio environment.

Procedure

1. Click the Open menu icon  and select **Content**.
2. Select the **External content** tab.
3. Navigate to a Watson Studio notebook and click its name.
4. In the **Save as** window, enter a name for the notebook that you're importing.
5. Navigate to a location in **My content** or **Team content**.
6. Click **Save**.

Results

You or a colleague can now select a content cell from the imported notebook and add it to a report or a dashboard. For more information, see "Adding a Notebook widget" or "Including output from a notebook" in the *Cognos Analytics Notebook Guide*.

Connecting Watson Studio notebooks to Cognos Analytics

If you are running Cognos Analytics on Cloud or Cognos Analytics on Cloud Pak for Data, you can configure Watson Studio notebooks to connect to Cognos Analytics.

Once it is configured, your Watson Studio notebook can

- read data assets from Cognos Analytics
- write data to Cognos Analytics

If you are using Cognos Analytics on Cloud

To connect your notebook to Cognos Analytics, add these two lines to the notebook:

```
from ca_data_connector import CADATAConnector

CADATAConnector.connect({'url': 'CA_on_Cloud_URL'});
```

where *CA_on_Cloud_URL* is the URL that you use to connect to Cognos Analytics on Cloud.

If you are using Cognos Analytics on Cloud Pak for Data

To connect your notebook to Cognos Analytics, add this line to the notebook:

```
CADATAConnector.connect({'url': 'CA_URL_in_Cloud_Pak_for_Data'});
```

where *CA_URL_in_Cloud_Pak_for_Data* is the URL that you use to connect to your Cognos Analytics instance on Cloud Pak for Data.

Note: You don't need to include your credentials in the line above. Cloud Pak for Data will use the credentials from your current Notebook session.

Best practices for displaying notebook visualizations in dashboards

As a notebook author, you can use some coding best practices to ensure that notebook visualizations are properly displayed when they are rendered in dashboards.

Resizing a Bokeh visualization

For the dashboard to properly resize any Bokeh visualization, don't hardcode the visualization width and height elements in the notebook. Also, when creating any figure, plot, or column, use the parameter `sizing_mode='scale_width'`.

Here is an example of the correct coding practice:

```
...
from bokeh.plotting import figure, show

x = [1, 2, 3]
y = [1, 2, 3]

p = figure(sizing_mode='scale_width')
p.line(x, y)
```

```
show(p)
'''
```

The visualizations that are coded this way scale properly in the dashboard, maintaining their aspect ratio.

Reloading a Bokeh visualization after a browser refresh

For a dashboard to properly reload a Bokeh visualization after a browser refresh, include the Bokeh initialization statement `output_notebook()` in the applicable notebook output cells.

Here is an example of the correct coding practice:

```
'''
# Notebook cell 1
from bokeh.plotting import figure, show

x = [1, 2, 3]
y = [1, 2, 3]








p = figure(sizing_mode='scale_width')
p.line(x, y)

# Notebook cell 2
output_notebook()
show(p)
'''
```

Chapter 4. Adding a Notebook widget

If IBM Cognos Analytics for Jupyter Notebook is enabled in IBM Cognos Analytics, you can use the **Notebook** widget to add notebooks to a dashboard or story.





Procedure

1. Click the **Widgets** icon .
- If you don't see the **Widgets** icon, click the **Edit** icon  first.
2. To add a notebook from the **Widgets** tab, complete the following steps:
 - a) If you want to position the widget yourself, drag the **Notebook** icon  to the canvas. If you want the widget to fill the next available pane in the template, click the **Notebook** icon.
 - b) From the **Notebook** widget, click **Select a notebook**.
 - c) Select a notebook and click **Open**.
 - d) From the available cells, select one notebook cell to include in your dashboard.
3. To see when a **Notebook** widget was last refreshed, hover your cursor over the time stamp icon .
4. To hide the time stamp icon  from the **Notebook** widget, complete the following steps:
 - a) Click the **Notebook** widget, click the **Properties** icon , and then click **Selection**.
 - b) Click the **Show timestamp** toggle.
5. To select a different notebook cell for a **Notebook** widget, select the widget, and click the **Edit widget** icon  from the widget toolbar.

Chapter 5. Including output from a notebook

If IBM Cognos Analytics for Jupyter Notebook is enabled in IBM Cognos Analytics, you can embed the output from a notebook code cell in a report.

Procedure

1. In edit mode, click the **Toolbox**  icon, expand **ADVANCED**, and then double-click  **Notebook**.
2. If you are in **Page design** view, double-click **Double click here to select a notebook cell**.
3. If you are in **Page preview** view, click **Select a notebook**.
4. Select a notebook, and then click **Open**.
5. Select one of the notebook cells, and click **OK**.
6. If you are in **Page design** view, switch to **Page preview** view to see the contents of the notebook cell.
7. To see when the notebook cell was last updated, hover your cursor over the time stamp  icon.
8. You can hide or show the time stamp  icon by doing the following steps:
 - a) Click the notebook cell.
 - b) In the **Properties** pane, under **Miscellaneous**, toggle the **Show timestamp** property.

Results

Note: When you run the report, the output is updated from the last time the notebook was run. Notebook output appears only in HTML output in the interactive viewer.

Chapter 6. Jupyter notebook samples

The notebook samples in this section are designed to provide ideas for using Jupyter notebooks with IBM Cognos Analytics.

Note: The samples covered in this topic are included with the Base Samples during a Cognos Analytics installation. After product installation, you can find them in the *installation_location/samples/notebooks* folder.

IBM will provide support if the asset does not work as described for the product version(s) identified. However, we are unable to provide custom support for you such as adding features or troubleshooting environmental issues. These will be logged in our system as future Feature requests. These assets are functional examples. They are code samples with narrow requirements and a specific use case, providing a baseline. They do not include every possible feature/interactivity or environment/configuration. You can use these samples as-is (supported by IBM), or you can modify/extend these samples to suit your business needs (not supported).

Flexible lightweight ETL notebook sample

This sample demonstrates ETL tasks within Cognos Analytics for Jupyter Notebook.

This sample notebook demonstrates how to conduct lightweight ETL (extract, transform, and load) tasks by connecting to an external data asset, performing some data cleansing, and writing the data into a Cognos Analytics asset.

This sample can be found here: Team content > Samples > Notebooks > Flexible lightweight ETL

Time series analysis notebook sample

Use an external CSV file in Cognos Analytics for Jupyter Notebook.

This sample notebook demonstrates how to connect to an external CSV file, display the data, write the data as a Cognos Analytics asset, modify the presentation of the data, and create a visualization.

Team content > Samples > Notebooks > IoT time series analysis

Visualization creation notebook sample

This sample notebook demonstrates how to connect to a local data file and create several visualizations (bar, column, line, pie, and bubble).

Sample can be found in Team content > Samples > Notebooks > Visualization notebook.

Retailer dashboard notebook sample

This sample demonstrates the integration of notebooks and dashboards.

Sample report can be found here: Team content > Samples > Notebooks > Retailer dashboard

Telecom data analysis notebook sample

This sample notebook demonstrates how to perform customer churn analysis on a sample dataset. Predictive analytics models are used to predict customer churn by evaluating their probability of risk to churn.

This sample can be found here: Team content > Samples > By feature > Notebooks > Notebooks.

Telecom data visualizations notebook sample

This sample notebook demonstrates how to explore data and create visualizations in the context of a fictional telecommunications company.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Notebooks](#)

Unit infection data notebook sample

This sample notebook demonstrates how to create a dataset and use visualization with a slider to display the rates of infection in a group of fictional hospital units over a given time period.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Notebooks](#).

Data quality template notebook sample

This notebook template can be applied to any dataset. It demonstrates how to read, review, quality check, and clean data.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Notebooks](#).

Schedule data creation notebook sample

This sample notebook demonstrates patient visit and nurse shift data at a fictional hospital.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Notebooks](#).

Schedule data creation notebook widget sample

This sample report demonstrates how to add notebook widgets to a report. These widgets display fictional health insurance coverage data from a notebook.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Reports](#).

Health insurance coverage analysis notebook sample

This sample notebook demonstrates how to create and merge geographic data with other data, modify the presentation of the data, and create map and trend line visualizations.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Notebooks](#).

Retailer report notebook widget sample

This sample report demonstrates how to add notebook widgets to a report. These widgets display fictional health insurance coverage data from a notebook.

This sample can be found here: [Team content > Samples > By feature > Notebooks > Reports](#).

