



Python Data Functions in TIBCO Spotfire®

Software Release 12.3.0

Contents

Python Packages in TIBCO Spotfire.....	3
Included Packages.....	5
Manage packages through roles.....	6
Developer role.....	6
Curator role.....	6
Administrator role.....	7
Creating an SPK for Python Packages.....	7
Removing a package from a deployment.....	9
SPK Versioning.....	11
Tips and Tricks for Working with Python in Spotfire.....	14
Spotfire and Python data type mapping.....	14
Troubleshooting Downloading Python Packages from a Proxy or Firewall Protected Installation.....	15
Troubleshooting "Empty String" errors in SBDF exports.....	16
Using an Alternative Python Package Repository.....	16
Tips for using a different Python interpreter.....	17
Read or Write Table and Column Metadata.....	17
Return a Graphic to Spotfire from Python.....	18
Clear the pyplot Image.....	18
Handling Extension dtypes Int32 and Int64.....	18
Integer Conversion.....	19
Using a Model Produced in A Previous Data Function.....	19
TIBCO Documentation and Support Services.....	20
Legal and Third-Party Notices.....	21

Python Packages in TIBCO Spotfire

When you create an analysis using Spotfire®, you can enhance the analysis by adding your own calculations and outputs using the data function framework. You can write data function scripts using different programming languages. This guide provides help for installing Python packages to use in data functions.

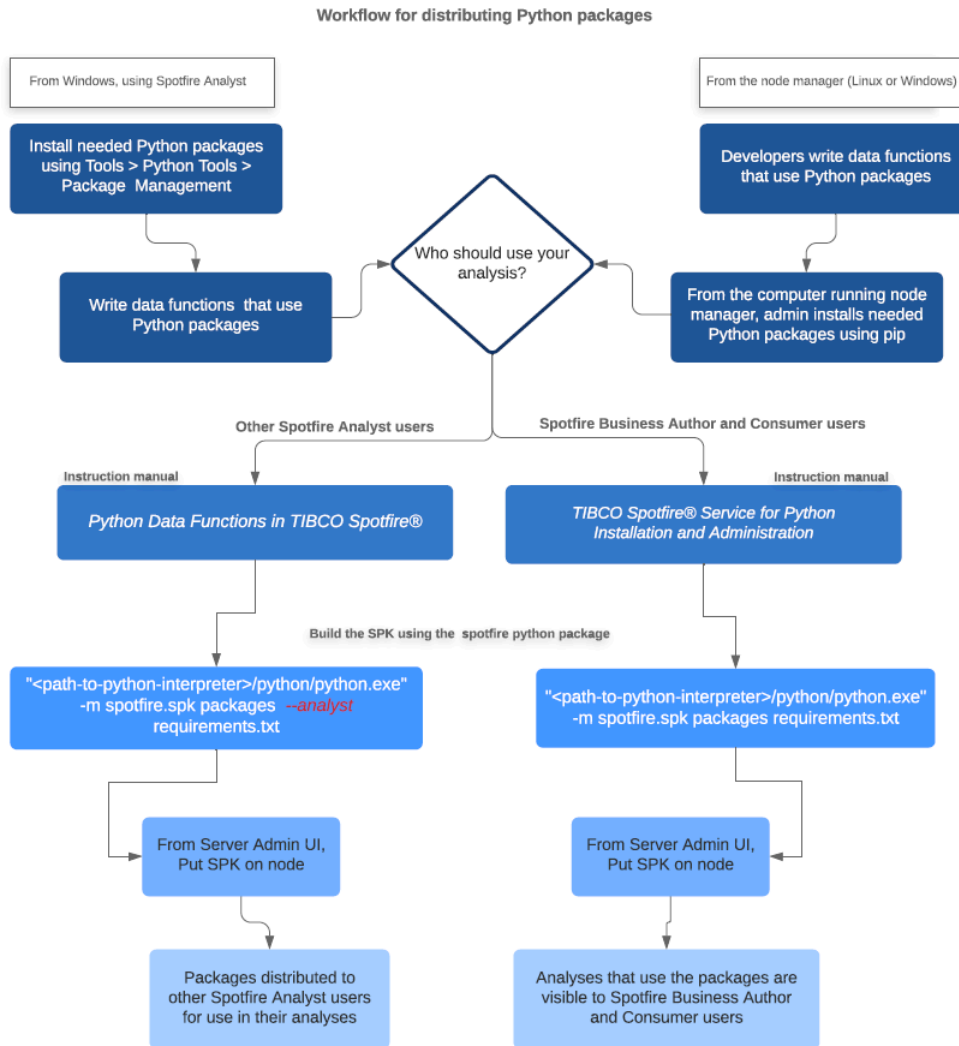
As of version 10.10 of TIBCO Spotfire®, you can develop Python data functions for your local Spotfire Analyst installation (as described in this document), or you can develop Python data functions to give to your server administrator to be deployed on a Spotfire Server for use from a web browser. See [TIBCO Spotfire® Service for Python](#) for more information.



To deploy an SPK to Spotfire Server containing packages for use with Python data functions that users access from a web browser, you must have TIBCO Spotfire® Service for Python, which is sold under a separate license.

Using Python data functions is different from the IronPython functionality available for extending Spotfire. For more information about using IronPython with Spotfire, see the TIBCO Spotfire® Analyst Help, available from the Spotfire menu, or the article [IronPython Scripting in TIBCO Spotfire®](#) on the TIBCO® Community web site.

The following workflow diagram provides a quick overview of determining which documentation you need (according to your role), who is going to use your packages, and which function to call to create an SPK for your target audience. This documentation, *Python Data Functions in TIBCO Spotfire®*, covers development and use of Python data functions on Spotfire Analyst installations.



This version of Spotfire includes a Python interpreter. For more information about using another installation of Python with your data functions, see the *TIBCO Spotfire® Help*.

You can download packages from the most popular Python repository web site PyPI, from another community repository, from a locally-managed repository, or from a package you created.

- If you are doing ad-hoc analysis and want to try a package from PyPI, you can use the Python Tools included in Spotfire for installing packages.

1. From the Spotfire menu, click **Tools > Python Tools > Package Management**.
2. Type the name of the package to install in the **Available Packages** text box, and then click **Search**.

For more information, see the *TIBCO Spotfire® User Guide*, from the **Help** menu in Spotfire.

- If your packages are managed by your Spotfire Server administrator, then you can create a Spotfire package (SPK) containing packages downloaded from PyPI, from another repository, or from a file location. See [Creating an SPK for Python Packages](#) on page 7 for more information.



Package versions installed using the SPK mechanism take precedence over packages installed using Python Tools. If you try to update a package that was previously installed by the Administrator, the updated package is installed, but the update is ignored.


If you share downloaded packages, you must ensure that the package versions are the same. For more information about managing packages through roles, see [Curator role](#) on page 6.

Included Packages

Your installation of Spotfire includes version 3.10.4 of the Python interpreter and several packages it needs to run under Spotfire. To run the Python interpreter, you must have the 64-bit version of Spotfire.



The packages listed in this table are required for Spotfire and the Python interpreter to work together. Removing or altering these packages can cause your Python data functions to fail.

Package name	Version	Description	More information
bitstring	4.0.1	A pure Python module that makes the creation, manipulation and analysis of binary data as simple and natural as possible.	https://pypi.org/project/bitstring/4.0.1/
numpy	1.24.1	<p>Provides the following.</p> <ul style="list-style-type: none"> • An N-dimensional array object. • Broadcasting functions. • Tools for integrating C/C++ and Fortran code. • Linear algebra, Fourier transform, and random number capabilities. <div>  <p>numpy version 1.19.4 has known incompatibilities when it is run on certain more recent versions of Windows 10. These compatibilities can cause Python to fail when it is used with Spotfire. If you build custom packages SPK on Windows, ensure the SPK does not include numpy version 1.19.4.</p> </div>	https://pypi.org/project/numpy/1.24.1
pandas	1.5.2	Provides data structures and data analysis tools for dealing with tabular data, ordered and unordered time series data, matrix data, and other types of data sets.	https://pypi.org/project/pandas/1.5.2/
pip	22.3.1	Provides support for installing packages.	https://pypi.org/project/pip/22.3.1
python-dateutil	2.8.2	Provides extensions to the datetime module in Python.	https://pypi.org/project/python-dateutil/2.8.2
pytz	2022.7	Provides a platform for cross-platform time zone calculations.	https://pypi.org/project/pytz/2022.7/

Package name	Version	Description	More information
setuptools	65.6.3	Provides tools for building, installing, upgrading, and uninstalling Python packages.	https://pypi.org/project/setuptools/65.6.3/
spotfire	1.9.0	Provides functions for integrating Python with Spotfire.	https://pypi.org/project/spotfire/
six	1.16.0	Provides utility functions for smoothing over the differences between the Python versions 2 and 3.	https://pypi.org/project/six/1.16.0/
wheel	0.38.4	The reference implementation of the Python wheel packaging standard, as defined in PEP 427 .	https://pypi.org/project/wheel/0.38.4/



* Exporting an SBDF that contains empty String columns causes an error with Pandas and Numpy. See [Troubleshooting "Empty String" errors in SBDF exports](#) on page 16 for more information.

Manage packages through roles

Working with packages in a deployment that includes Spotfire and Spotfire Server can add some complexity to management policies.

The job of synchronizing package versions among your development computers, your testing computers, and your servers is an important package management concern for an organization. You can reduce the risk of confusion and streamline your processes by defining roles in your organization for dealing with packages. Ensure processes and rules are established to manage packages.

Developer role

The developer is a Python programmer or statistician who develops packages or writes data functions using Python.

The package developer accomplishes the following tasks using the Spotfire tools.

- Develops and tests Python packages or data functions using the local Python interpreter available from Spotfire Analyst.
- Reviews and recommends packages to be included on the Spotfire Server for data functions to use.

Curator role

The curator maintains the standards and lists of officially-sanctioned packages. The curator keeps all of the package versions synchronized. The curator might be the same person who fills the developer role.

The approval process for adding a packaging is up to your organization, and might vary from minimal to extensive, depending on your usual practices. Designate a developer familiar with Python packages and package versioning to be the package curator. The package curator works with package developers and server administrators to perform the following management tasks.

- Maintains the list of tested and sanctioned package versions (the gold standard), which would be the set of packages available for general use under Spotfire applications.
- Creates a Spotfire SPK containing the Python packages, and then gives it to the administrator who manages Spotfire distributions on Spotfire Server. Packages uploaded to Spotfire Server are distributed to other Python users who write data functions for using Python in Spotfire Analyst.

- Ensures that the SPK containing the "gold standard" package versions are placed on the Spotfire Server, to be distributed to Spotfire analysts who develop data functions, and who need to use the same packages.
- Optionally, ensures that the SPK containing these packages are placed on Spotfire Server to be used by [Spotfire Service for Python](#) for Spotfire Business Author and Consumer users.
- Python package versions shared among team members must be kept synchronized.
- You can install multiple SPKs containing Python packages on the Spotfire Server, as long as each SPK has a unique name and ID.
- Uploading a new SPK overwrites any older version of that same SPK that was previously deployed.



See [SPK Versioning](#) on page 11 for more information.

Administrator role

The Spotfire administrator manages packages on the Spotfire Server.

The responsibilities for the administrator role include the following.

- Deploys the SPK containing Python packages to be called by TIBCO Spotfire® Service for Python or that are distributed to Spotfire Analyst users. For more information about Spotfire® Service for Python, see its documentation at <https://docs.tibco.com/products/tibco-spotfire-service-for-python>.
- Assigns licenses for access to the Data Functions feature in Spotfire Analyst.
- Uploads, maintains, and removes packages. (Might assign server permissions to the curator for this task.)

Creating an SPK for Python Packages

Your installation of Spotfire Analyst includes a Python interpreter and a set of packages to enable using Python in Spotfire. One of these packages, `spotfire`, provides tools for building SPKs to share Python packages with other data function authors in your organization, or with Spotfire Service for Python on the Spotfire Server.

Spotfire Analyst relies on `pip`, the Python command-line application for Python package installation. Spotfire Analyst uses a `requirements.txt` file to specify the packages to include in your SPK.

By default, the file `requirements.txt` searches the PyPI package site for the specified package and version.

- To include a package from a different repository or in a local file path, in the `requirements.txt` file, use the option `-i` or `--index-url`, followed by the location URL.

```
#example
#
mylib -i http://my.domain.org/lib/1.0.0/mylib/
```

- To include a `.whl` package, in the `requirements.txt` file, provide the relative path to the package from the current working directory.

```
#simple-example
#
./my_path/my_package.whl
packaging==1.0.0
```

For more information about creating a `requirements.txt` file for your package list, see its documentation at the following location.

- https://pip.readthedocs.io/en/stable/user_guide/#requirements-files



- https://pip.readthedocs.io/en/stable/reference/pip_install/#requirements-file-format

Perform this task from a command prompt on the Windows computer where Spotfire Analyst is installed.



Your installation of Spotfire Analyst relies on the Python packages included in the installation. Removing any of these packages causes your Spotfire Analyst installation to not work with the included Python interpreter. See [Included Packages](#) on page 5 for more information.

Prerequisites

- You must have the appropriate Spotfire license for authoring data functions.
- If you are using a Python interpreter other than the one provided with your Spotfire installation, then you must first run the following command:

```
-m pip install spotfire
```



This command downloads and installs the spotfire package, which is included with the Python interpreter provided with your Spotfire installation, and which is required to use Python with Spotfire.

- If you are creating an SPK for Spotfire Business Analyst users (from a web browser), your organization must have a license for Spotfire Service for Python, and that service must be installed on a node accessible to Spotfire Server. (See [Spotfire Service for Python](#) for more information.)
- You must have created the file `requirements.txt` containing the list of packages to include in your SPK.

The following example specifies these packages and specified versions from PyPI.

```
#
##### example-requirements.txt #####
#
scipy == 1.8.0
matplotlib == 3.5.2
statsmodels == 0.13.2
```


Procedure

1. From the command line, type the following command.



For this command, you must quote the path string, as shown in the following example.

```
"%SPOTFIRE_HOME/Modules/Python Interpreter_<version#>
/python/python.exe" -m spotfire.spk packages
[--name "<package-name>"]
[--analyst] <name.spk>
<path-to>requirements.txt
```

Option	Description
spotfire.spk	<p>The package containing the Python code to download and bundle the needed packages specified in <code>requirements.txt</code>.</p> <div>  <p>From the command prompt, you can view help for the spotfire package by typing the following command:</p> <pre>"<path-to-python-interpreter>\python\python.exe" -m spotfire.spk packages --help</pre> </div>
packages	The subcommand that creates the package bundle.

Option	Description
<code>--name</code>	<p>A string containing the friendly name of the SPK to contain the packages.</p> <p>This value is displayed in the node manager. If you provide no value for this parameter, then the package name defaults to whatever was last used in the brand located in the <code>requirements.txt</code> file, or if no brand exists, in the current "Python Packages". Additionally, the package ID defaults to the current value from the brand in the <code>requirements.txt</code> file, or a random UUID if none exists.</p>
<code>--analyst</code>	<p>Specifies that the Spotfire Server should distribute the packages in the SPK to other Spotfire Analyst clients connected to the Spotfire Server.</p> <p>(Do not use this option if you want to install an SPK for Spotfire Service for Python to use.)</p>
<code>name.spk</code>	The name of the SPK file that is created by this task.
<code>requirements.txt</code>	The full path to the file <code>requirements.txt</code> .

The following example creates an SPK named `my-pkgs.spk`, containing the packages specified in `requirements.txt`.

```
"%Spotfire_Home%\Modules\Python Interpreter_3.10.4.0\python\python" -m spotfire.spk
packages --name "my-pkgs.spk --analyst C:\files\requirements.txt
```

The packages and all of their dependencies are written to the SPK named `my-pkgs.spk` in the current working directory where the command was run, and the version information is recorded in the file `requirements.txt`. For example:

```
#
##### example-requirements.txt #####
#
scipy == 1.8.1
matplotlib == 3.5.2
statsmodels == 0.13.2
## spotfire.spk: {"BuiltBy":"3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022,
## spotfire.spk: 23:13:41) [MSC v.1929 64 bit (AMD64)]","BuiltAt":"Fri M
## spotfire.spk: ay 27 13:35:31 2022","BuiltFile":"sanctioned-packages.s
## spotfire.spk: pk","BuiltName":"sanctioned-packages","BuiltId":"5ac779
## spotfire.spk: d6-8866-4a4d-bbec-af7b46e8a86a","BuiltVersion":"1.0.0.0
## spotfire.spk: ","BuiltPackages":{"cyclers":"0.11.0","fonttools":"4.33.
## spotfire.spk: 3","kiwisolver":"1.4.2","matplotlib":"3.5.2","packaging
## spotfire.spk: ":"21.3","patsy":"0.5.2","pillow":"9.1.1","pyparsing":"
## spotfire.spk: 3.0.9","scipy":"1.8.1","statsmodels":"0.13.2"}}
```

2. Locate the SPK you created.

This file is saved in the directory from which you ran the command.

3. Give the SPK to the Spotfire Server administrator to deploy.

Result

After the SPK is placed into the deployment area, if you specified the `--analyst` flag, then Spotfire Analyst users connected to that deployment area are prompted to update their Spotfire Analyst installations. The packages are included in the update.

Removing a package from a deployment

In some cases, you might need to remove a Python package from a deployment. You must recreate and redeploy the SPK.

Prerequisites

- You must have the appropriate Spotfire license for authoring data functions.
- If you are using a Python interpreter other than the one provided with your Spotfire installation, then you must first run the following command:

```
-m pip install spotfire
```



This command downloads and installs the spotfire package, which is included with the Python interpreter provided with your Spotfire installation, and which is required to use Python with Spotfire.

- You must create a new `requirements.txt` file containing the list of packages to keep in the SPK.

The following example shows three packages from the example for creating the original SPK.

```
#
##### example-requirements.txt #####
#
scipy == 1.8.1
matplotlib == 3.5.2
statsmodels == 0.13.2
```



For information about how changing a `requirements.txt` affects how the SPK version changes, see [SPK Versioning](#) on page 11.

Procedure

- To remove one of the packages, recreate the `requirements.txt` file, removing the package name to delete.



Save a backup copy of your `requirements` file before altering it.


```
#
##### example-requirements.txt #####
#
scipy == 1.8.1
statsmodels == 0.13.2
```

- From the command line, type the following command.



For this command, you must quote the path string, as shown in the following example.

```
"<path-to-python-interpreter>/python/python.exe" -m spotfire.spk packages
--analyst <name.spk>
<path-to>requirements.txt
```

Option	Description
spotfire.spk	<p>The package containing the Python code to download and bundle the needed packages specified in <code>requirements.txt</code>.</p> <div>  <p>From the command prompt, you can view help for the spotfire package by typing the following command:</p> <pre>"<path-to-python-interpreter>/python/python.exe" -m spotfire.spk packages --help</pre> </div>
packages	The subcommand that creates the package bundle.

Option	Description
<code>--analyst</code>	Specifies that the Spotfire Server should distribute the packages in the SPK to other Spotfire Analyst clients connected to the Spotfire Server. (Do not use this option if you want to install an SPK for Spotfire Service for Python to use.)
<code><name.spk></code>	The name of the SPK to contain the packages.
<code>requirements.txt</code>	The full path to the <code>requirements.txt</code> .

The following example creates an SPK named `my_pkgs.spk`, containing the packages specified in `requirements.txt`, (which no longer contains the package to delete), and any of their dependencies.

```
"%Spotfire_Home%\Modules\Python Interpreter_3.10.4.0\python\python" -m spotfire.spk
packages --analyst my-pkgs.spk c:\files\requirements.txt
```

The packages and all of their dependencies are written to the SPK named `my-pkgs.spk`, and the version information is recorded in the file `requirements.txt`. For example:

```
#
##### example-requirements.txt #####
#
scipy == 1.8.1
statsmodels == 0.13.2
## spotfire.spk: {"BuiltBy":"3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022,
## spotfire.spk: 23:13:41) [MSC v.1929 64 bit (AMD64)]","BuiltAt":"Fri M
## spotfire.spk: ay 27 13:41:54 2022","BuiltFile":"sanctioned-packages.s
## spotfire.spk: pk","BuiltName":"sanctioned-packages","BuiltId":"5ac779
## spotfire.spk: d6-8866-4a4d-bbec-af7b46e8a86a","BuiltVersion":"2.0.0.0
## spotfire.spk: ","BuiltPackages":{"packaging":"21.3","patsy":"0.5.2","
## spotfire.spk: pyparsing":"3.0.9","scipy":"1.8.1","statsmodels":"0.13.
## spotfire.spk: 2"}}
```

3. Locate the SPK you created.

This file is saved in the directory from which you ran the command.

4. Give the SPK to the Spotfire Server administrator to deploy.

Result

After the SPK is placed into the deployment area, if you specified the `--analyst` flag, then Spotfire Analyst users connected to that deployment area are prompted to update their Spotfire Analyst installations. Only the specified packages are included in the update, and the package removed from the `requirements.txt` file is deleted from users' computers.

SPK Versioning

To share packages among data function authors in your organization, you can create the file `<your-filename>.spk` containing the packages to distribute to others, or to Spotfire Service for Python running on the Spotfire Server. You might need to change or update the packages or package versions that you distribute, which requires changing the version of the SPK containing the packages.


You can create or change a Spotfire Server SPK using the steps described in [Creating an SPK for Python Packages](#) on page 7. The package `spotfire.spk` creates a new SPK using the versioning rule details for the following tasks.



- Python package versions shared among team members must be kept synchronized.
- You can install multiple SPKs containing Python packages on the Spotfire Server, as long as each SPK has a unique name and ID.
- Uploading a new SPK overwrites any older version of that same SPK that was previously deployed.

The SPK property `BuiltVersion` is NOT the same as the package version. `BuiltVersion` is always specified as four components (*N.n.n.n*).

Versioning rules

Task	Example	Notes and Results
Create a new <code>requirements.txt</code> .	<p>From a text editor, create the file <code>requirements.txt</code> containing the names and version numbers of the packages to include. For example,</p> <pre># ##### example-requirements.txt ##### # # scipy == 1.8.0 matplotlib == 3.5.2 statsmodels == 0.13.2</pre> <p>From a command prompt, set the Python path and issue the command to create the SPK. For example, the following command creates an SPK named "Authorized Python Packages" to be distributed to Spotfire Analyst users.</p> <pre>"%SPOTFIRE_HOME\Modules\Python Interpreter_\python\python.exe" -m spotfire.spk packages --name " Authorized Python Packages" --analyst requirements.txt</pre>	<p>The SPK version containing the three specified packages, along with their required and dependent packages, is created, and the <code>BuiltVersion</code> property values is set to 1.0.0.0. The script overwrites any pre-existing SPK of the same file name.</p> <p> If you use this method to overwrite an existing SPK, the version number is still set to 1.0.0.0; therefore, Spotfire Server does not register the package as a new one, so it does not distribute the included packages to the users.</p> <p>You can see the <code>BuiltVersion</code> and the dependent/required additional packages installed by opening the <code>requirements.txt</code> in a text editor.</p> <p>For example:</p> <pre>"BuiltVersion": "1.0.0.0"</pre>
Add packages to an existing SPK to put on Spotfire Server.	<p>From a text editor, open the <code>requirements.txt</code> that you used for your previous deployment of your SPK, and add the name and version number of the package to include. Save the text file.</p> <p>From a command prompt, set the Python path and issue the command to create the SPK. For example, the following command creates an SPK named "Authorized Python Packages" to be distributed to Spotfire Analyst users.</p> <pre>"%SPOTFIRE_HOME\Modules\Python Interpreter_\python\python.exe" -m spotfire.spk packages --name " Authorized Python Packages" --analyst requirements.txt</pre>	<p>The SPK <code>BuiltVersion</code> is incremented by a minor number, and the new package and its dependent packages are added. For example:</p> <pre>"BuiltVersion": "1.1.0.0"</pre>

Task	Example	Notes and Results
Remove a package from the existing SPK (and subsequently from Spotfire Server).	<p>From a text editor, open the <code>requirements.txt</code> that you used for your previous deployment of your SPK, and delete the name and version number of the package to remove. Save the text file.</p> <p>From a command prompt, set the Python path and issue the command to create the SPK. For example, the following command creates an SPK named "Authorized Python Packages" to be distributed to Spotfire Analyst users.</p> <pre data-bbox="485 485 1023 625">%SPOTFIRE_HOME/Modules/Python Interpreter_/python/python.exe" -m spotfire.spk packages --name " Authorized Python Packages" --analyst requirements.txt</pre>	<p>The SPK <code>BuiltVersion</code> is incremented to the next major number, and the package you deleted and its dependent packages are removed. For example:</p> <pre data-bbox="1054 338 1481 390">"BuiltVersion": "2.0.0.0"</pre>
Assigning a specific <code>BuiltVersion</code> number for the SPK.	<p>From a command prompt, set the Python path and issue the command, but this time specify the <code>BuiltVersion</code> to use by passing it to the <code>version</code> argument. For example, the following command creates an SPK named "Authorized Python Packages" with the <code>BuiltVersion</code> number set to 1.2.3.4 .to be distributed to Spotfire Analyst users.</p> <pre data-bbox="485 863 1023 1003">%SPOTFIRE_HOME/Modules/Python Interpreter_/python/python.exe" -m spotfire.spk packages --name " Authorized Python Packages" -v=2.3.4.5 --analyst requirements.txt</pre>	<p>The SPK <code>BuiltVersion</code> is set to the value passed in for the <code>-v</code> argument. For example:</p> <pre data-bbox="1054 737 1481 789">"BuiltVersion": "2.3.4.5"</pre>

Tips and Tricks for Working with Python in Spotfire

You can expand the functionality of Spotfire using Python data functions, but innate differences in these two tools can cause unexpected behavior.

This section contains some tricks for working with graphic types, models, and data types between Python and Spotfire.

Spotfire and Python data type mapping

To create Python data functions in Spotfire, you need to know how the data types in each application map to each other. These tables provide that mapping, including data type mapping to Pandas column dtype, and for mapping columns and tables.

Exported data from Spotfire, received by Python

Data Type Exported from Spotfire	Data Type Received By Python	Pandas column dtype
Integer	int	Int32
LongInteger	int	Int64
Real	float	float64
SingleReal	float	float32
Currency	decimal.Decimal	object
Date	datetime.date	object
DateTime	datetime.datetime	object
Time	datetime.time	object
TimeSpan	(not supported)	(not supported)
Boolean	bool	object
String	str	object
Binary	bytes	object

- Spotfire columns map in Python to Pandas `Series` type.
- Spotfire tables map in Python to Pandas `DataFrame` type.



* Exporting an SBDF that contains empty String columns causes an error with Pandas and Numpy. See [Troubleshooting "Empty String" errors in SBDF exports](#) on page 16 for more information.

Sent from Python and Imported into Spotfire

Data Type Sent By Python	Data Type Imported Into Spotfire
int	LongInteger

Data Type Sent By Python	Data Type Imported Into Spotfire
float	Real
decimal.Decimal	Currency
datetime.date	Date
datetime.datetime	DateTime
datetime.time	Time
datetime.timedelta	TimeSpan
bool	Boolean
str	String
byte	Binary

Troubleshooting Downloading Python Packages from a Proxy or Firewall Protected Installation

If the computer you are using to download Python packages for use in Spotfire is behind a firewall or runs through a proxy service, then you can encounter the error "Package repository not accessible" "Python Package Index is not accessible."

To troubleshoot this error, work through the steps in this topic. Installing a package using the Spotfire point-and-click Python Tools Package Management dialog box option calls the underlying Python pip (package management) command. The pip command has no built-in capability to auto-configure proxy settings, so when the command encounters a proxy, the command fails.

Perform this task on the computer where Spotfire Analyst 12.3.0 or later is installed.

Prerequisites

- Browser access to the URL <https://pypi.org/>.

Procedure

1. From the Spotfire menu, click **Tools > Python Tools**.
2. In the Python Tools dialog box, click **Copy Python Interpreter Path to Clipboard**.
3. From the Windows **Start** menu, open a Windows command prompt (cmd).
4. At the command prompt run the following command.

```
<pasted path to Python> -m pip search pip
```

for example, the complete line to execute should look like the following.

```
"C:\Program Files\TIBCO\Spotfire\12.3.0\Modules\Python Interpreter_3.10.4.0\python\npython.exe"  
-m pip search pip
```

If this command fails, then (for a single computer configuration) change the pip configuration as follows.

- At the Windows command prompt, type the following command.

```
<pasted path to Python> -m pip config set global.trusted-host pypi.org
```

for example, for a Spotfire 12.3.0 installation, the complete line to execute should look like the following.

```
"C:\Program Files\TIBCO\Spotfire\12.3.0\Modules\Python Interpreter_3.10.4.0\python\npython.exe"  
-m pip config set global.trusted-host pypi.org
```

Result

You should now be able to install Python packages.

What to do next

Review the guidelines at https://pip.pypa.io/en/stable/user_guide/#configuration for information about setting the parameter for multiple users.

Troubleshooting "Empty String" errors in SBDF exports

When you export an SBDF from Python (for example, as output for a data function), and your output contains an empty column, you can encounter a data type error.

Error

The error looks like this:

```
spotfire.sbdf.SBDFError: cannot determine type for column 'EmptyString'; all values are missing
```

This error happens because the Spotfire data function environment cannot determine the proper Spotfire type to export the data as if all values in the column are missing (in other words, Python's None, NumPy's nan, or Panda's NA or NaT values).

Resolution

To resolve this error, edit your data function to use the helper function `set_spotfire_types`.

Example

```
import pandas as pd  
import spotfire  
  
df = pd.DataFrame(inp)  
spotfire.set_spotfire_types(df, {'EmptyString': 'String'})
```

Using an Alternative Python Package Repository

If your company keeps an internal repository for approved Python packages, you can set a property in Spotfire Administration Manager to point to the URL for the internal repository.

Changing this property applies only to installing packages using the Spotfire Analyst feature **Tools > Python Tools > Package Management**. If you are installing packages using the `pip` command, and you want to use your internal repository, then pass the URL for your internal repository in the `pip` command parameter `--index-url`.



Packages that are provided with Spotfire Service for Python are not affected by this setting.

Prerequisites

You must have a Spotfire Administration license to access the Administration Manager tools.

Procedure

1. In Spotfire Analyst, click **Tools > Administration manager**.
The Administration Manager dialog box is displayed.
2. Click the **Preferences** tab.
3. From the **Selected group** list, click the name of the group for which to set the property.
For example, from the list select **Script Author** to set the preference for any user with the Script Author license. To change the value for all users, select **Everyone**.
The Preferences list for the selected group name is displayed.
4. In the resulting list box, expand **DataFunctions**, and then click **DataFunctionsPreferences**.
5. Click **Edit**.
The properties for **DataFunctionsPreferences** are now editable.
6. From the property list, select **PythonPackageRepositoryIndexURL** and then, in the right column for this property, provide the URL for your internal repository.

Result

The custom URL overrides the default PyPI package index and accesses only the packages available in the specified custom internal repository.

What to do next

Repeat this process for any other groups that author data functions using the Spotfire Service for Python. For more information, see the help for Administration Manager.

Tips for using a different Python interpreter

If you have another Python interpreter installed on the computer where Spotfire is installed, you can specify using that interpreter with Spotfire.

From the **Tools > Options > Data Functions** dialog box, you can select **Use specific local python.exe**, and then provide an alternative path to your Python interpreter. If you use a different interpreter, make sure that you consider the following.

- The Python interpreter must be version 3.7 or later.
- Any packages you install are installed in the `site-library` directory for that interpreter.
- You must install the packages required by Spotfire to work with Python. See [Included Packages](#) on page 5 for the list.

Read or Write Table and Column Metadata

When you read or write a Pandas data frame, table and column metadata is provided to help you ensure that the content type is set correctly.

Two attributes are provided that make the metadata accessible:

- `spotfire_table_metadata`
- `spotfire_column_metadata`



Table metadata applies only to tables, and column metadata applies only to columns.

Table and column names are prefixed to the metadata names. The following examples illustrate the naming.

- If your input parameter is 'Table', and if your table name is MyTable, then the table attribute is `MyTable.spotfire_table_metadata`.
- If your input parameter is 'Table', and if a column name in MyTable is Cost, then that column metadata is `MyTable['Cost'].spotfire_column_metadata`.
- If your input parameter is 'Column', and if a column name is Cost, then that column metadata is `Cost.spotfire_column_metadata`.



Due to limitations in Pandas, metadata does not survive a Pandas operation that constructs a new data frame or series. If you try to access the metadata on a newly-created object, an `AttributeError` is raised.

Setting the `spotfire_table_metadata` on a Pandas `DataFrame` object usually triggers a warning from Pandas of the format "UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see <https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access>", which is reported through Spotfire as a warning when it runs the data function. You can silence the warning using the following construction.

```
import warnings
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    x.spotfire_table_metadata = {...}
```



Setting `spotfire_column_metadata` on Series objects does not trigger this warning.

You can copy metadata using the function `spotfire.copy_metadata(source, destination)`. This function is located in the top-level "spotfire" module, and can be imported into your script using the command `import spotfire`. This function copies all of the metadata from the first argument to the second argument, provided that both arguments are the same type (that is, both must be either `DataFrames`, or both must be `Series`). If the types differ, a `TypeError` is raised.

Return a Graphic to Spotfire from Python

Spotfire uses an image binary to display graphic images in a document property.

When you return a graphic from Python to be used in Spotfire, Spotfire automatically converts these image types into an image binary that it can use as a document property. This technique works for the following graphical formats.

- matplotlib figure objects.
- PIL image objects.
- seaborn Grid objects.

These formats cover the majority of the graphical content available in Python packages.

Clear the pyplot Image

A plot created in a data function using matplotlib's `pyplot` function can persist between data functions unless the plot is explicitly cleared.

If the buffer retains the plot, you can explicitly clear it by including the command `plt.clf()` in your data function.

Handling Extension dtypes Int32 and Int64

Third-party packages might not support the extension dtypes `Int32` and `Int64`.

The Python implementation in Spotfire uses the extension dtypes `Int32` and `Int64`. These dtypes support missing values in data coming from Spotfire.

However, some packages, including pandas-profiling, do not support these extension dtypes. If you use a package with these dtypes, you could encounter errors. For third-party packages to work as expected, try casting the columns that cause problems to `int32` or `int64`.

For example, for a simple table with one column of 64-bit integers, you can recast it as follows for your Python package.

```
MyTable = MyTable2.astype('int64')
```

For more information, see <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.astype.html>.

Integer Conversion

When a data function sends a single integer between Python and Spotfire, Spotfire can automatically promote the integer to a `longInteger`, causing the data function to fail if the output is set to an integer type property.

Because Python has only one integer type for single integers, and because Spotfire converts certain integers to long integers, you must pay attention to single integers passed between Python and Spotfire.

You can avoid this error by specifying ints as `longIntegers` in your data functions.

For more information about how to map data types, column types, and tables between Spotfire and Python, see [Spotfire and Python data type mapping](#) on page 14.

Using a Model Produced in A Previous Data Function

If you produce an object using a Python data function, and you want to bring it into another data function for further handling, the object might need special handling.

If you create an object, such as a machine-learning model, and you want to use that object in a second data function, you can use the pickle package to store the object in a document property. The following example demonstrates using the pickle package to manage this process.

```
#From data function number 1
import pickle
binary = pickle.dumps(myObject)
```

```
#To data function number 2
import pickle
myObject = pickle.loads(binary)
```

The binary data can be stored in a Spotfire document property in the first data function, and then loaded from that document property to the second data function. This technique should work generically for almost any object that you might want to retain between data function calls.

TIBCO Documentation and Support Services

For information about the Spotfire products, you can read the documentation, contact TIBCO Support, and join the TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The website is updated frequently and is more current than any other documentation included with the product.

TIBCO Spotfire Documentation

The documentation for all Spotfire products is available on the [TIBCO Spotfire® Documentation](#) page. This page takes you directly to the latest version of each document.

To see documents for a specific Spotfire product or version, click the link of the product under 'Other versions', and on the product page, choose your version from the top right selector.

Release Version Support

Some release versions of TIBCO Spotfire products are designated as long-term support (LTS) versions. LTS versions are typically supported for up to 36 months from release. Defect corrections will typically be delivered in a new release version and as hotfixes or service packs to one or more LTS versions. See also https://docs.tibco.com/pub/spotfire/general/LTS/spotfire_LTS_releases.htm.

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking **Register** on the website.

System Requirements for Spotfire Products

For information about the system requirements for Spotfire products, visit <http://spotfi.re/sr>.

How to join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

For quick access to TIBCO Spotfire content, see <https://community.tibco.com/products/spotfire>.

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, TIBCO Spotfire, TIBCO Spotfire Analyst, TIBCO Spotfire Automation Services, TIBCO Spotfire Server, TIBCO Spotfire Web Player, TIBCO Enterprise Runtime for R, TIBCO Enterprise Runtime for R - Server Edition, TERR, TERR Server Edition, and TIBCO Spotfire Statistics Services are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme.txt file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SOFTWARE GROUP, INC MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 1994-2023 Cloud Software Group, Inc. All Rights Reserved.