ScienceLogic Libraries and Execution Environments

SL1 version 12.2.0 (Document revision 1)

Table of Contents

Chapter

1

Introduction to ScienceLogic Libraries and Execution Environments

Overview

This manual describes how to create and manage ScienceLogic Libraries and Execution Environments.

This chapter covers the following topics:

What is a ScienceLogic Library?

Some Dynamic Applications, credential tests, and Run Book Actions in SL1 collect data or complete automated actions by executing one or more blocks of Python code, called *snippets*. When this happens, SL1 passes credentials and other configuration information to the snippet, and at the end of execution, the snippet either passes collected data back to SL1 or runs an action. Upon release of version 12.1.0, SL1 upgraded to Python 3.

A *ScienceLogic Library* is a package consisting of metadata and Python files that can be used by the Dynamic Applications, credential tests, and Run Book Actions that use snippets. ScienceLogic Libraries can be included in a PowerPack along with the Dynamic Applications, credential tests, and Run Book Actions that they support.

These are a few reasons you might want to create and use a ScienceLogic library in your PowerPack:

- You have a few short Python functions or classes you want to share between snippets (py_directory).
- You have a Python package you created that you want to share between snippets (py_package).
- You have a third-party package that needs to be available during snippet execution (py_package).

ScienceLogic Libraries have two formats. The type that you use depends on your Python expertise and your requirements:

- *py_directory (.tar format)*. This file type includes metadata and a directory of Python source files. This format is most useful for users who are not experienced Python developers or who do not want the overhead of Python packages.
- *py_package (.tar format)*. This file type includes metadata and Python packages. This format requires an understanding of Python packaging tools. While this format is more complex, it also allows you to more easily take advantage of unit tests, automatic versioning, and other common packaging practices.

ScienceLogic Libraries can be installed on SL1 in three ways:

- Some libraries are included when you install SL1 or run system updates.
- Libraries that are included in a PowerPack are installed on SL1 when you install the PowerPack.
- Libraries can be manually imported into SL1.

You can view all of the installed ScienceLogic Libraries on the **ScienceLogic Library Manager** page (System > Customize > ScienceLogic Libraries).

For details about how to manage libraries in SL1, see the *Managing ScienceLogic Libraries* chapter.

What is an Execution Environment?

To use a ScienceLogic Library, you must align it with an execution environment.

In the SL1 user interface, an **execution environment** is a list of one or more ScienceLogic libraries. When a Dynamic Application snippet, Run Book Automation snippet, or credential test is executed, the execution environment defines a virtual Python environment that is deployed on-demand and includes all of the ScienceLogic libraries aligned with it for use during snippet execution.

A ScienceLogic Library can be associated with multiple execution environments, allowing you to share code between snippets, Dynamic Applications, and PowerPacks. A ScienceLogic Library can be aligned with only a single execution environment at a time.

Execution environments must be specified when creating or editing the following:

- Snippets
- Credential tests
- Run Book Action policies that execute a snippet

When SL1 runs a snippet embedded in a Dynamic Application, credential test, or a Run Book Action, it automatically deploys the runtime environment defined by the execution environment. The runtime environment includes each library aligned with the execution environment, plus system libraries that are provided by SL1.

If you do not specify the execution environment for a Dynamic Application, credential test, or Run Book Action, SL1 aligns them with a system-default execution environment. This default execution environment is defined in each SL1 System update.

When you build a PowerPack, it automatically includes the execution environments that are aligned with the PowerPack's Dynamic Applications, credential tests, and Run Book Actions that execute snippets.

For details about how to create and manage execution environments in SL1, see the *Managing Execution Environments* chapter.

For more information about aligning execution environments with Dynamic Applications, credential tests, or Run Book Actions, see the following manuals:

- *Dynamic Application Development*
- *Snippet Dynamic Application Development*
- *Internal Collection Dynamic Application Development*
- *Discovery and Credentials*
- *Run Book Automation*

Chapter

2

Managing ScienceLogic Libraries

Overview

This chapter describes how to manage ScienceLogic Libraries in SL1.

This chapter covers the following topics:

Viewing the List of ScienceLogic Libraries

The **ScienceLogic Library Manager** page (System > Customize > ScienceLogic Libraries) displays a list of all available ScienceLogic Libraries. From this page, you can download libraries, import new libraries, delete existing libraries, or view and edit a list of Execution Environments that you can associate with a library.

For each library, the page displays the following information:

> **TIP:** To sort the list of ScienceLogic Libraries, click on a column heading. The list will be sorted by the column value, in ascending order. To sort by descending order, click the column heading again. The **Last Edited** column sorts by descending order on the first click; to sort by ascending order, click the column heading again.

- *Library Name*. The name of the ScienceLogic Library.
- *Version*. The version number of the ScienceLogic Library.
- *System*. Displays **True** if the ScienceLogic Library is included in the latest release of SL1 and is aligned with the "System" environment. Displays **False** if the ScienceLogic Library is included in an imported PowerPack or manually installed by an administrator.
- *Type*. The file type used to bundle and deliver the ScienceLogic Library. Supported types include the following:
    - *py_directory (.tar format)*. A py_directory is a tar file that includes Python files and is typically used by novice developers to share short code snippets between Dynamic Applications or SL1 stacks. This format does not let you manage items such as dependencies, version, or test, and is not recommended.
    - *py_package (.tar format)*. A py_package is a tar file that contains Python packages and metadata files (name, version, description, etc.). The py_package format is typically used by more experienced developers to build complex solutions. ScienceLogic recommends using the py_package format because it is easier to test and maintain.
- *Dependencies*. A comma-separated list of other ScienceLogic Libraries on which the selected library is dependent, if applicable. These additional libraries are included in the selected library's *py_package.tar* file.
- *Description*. A single-line description of the ScienceLogic Library's intended use.
- *Python Version*. The Python compatibility version.
- *Edited By*. The name of the ScienceLogic user who created or last edited the ScienceLogic Library.
- *Last Edited*. The date and time the ScienceLogic Library was created or last edited.

Searching and Filtering the List of ScienceLogic Libraries

The Filter-While-You-Type fields appear as a row of blank fields at the top of the list. These fields let you filter the items that appear in the list.

The list is dynamically updated as you select each filter. For each filter, you must make a selection from a drop-down menu or type text to match against. SL1 will search for entries that match the text, including partial matches. Text matches are not case-sensitive, and you can use special characters in each text field.

By default, the cursor is placed in the first Filter-While-You-Type field. You can use the <Tab> key or your mouse to move your cursor through the fields.

You can filter by one or more of the following parameters. Only items that meet all of the filter criteria are displayed on the page.

- *Library Name*. You can enter text to match, including *special characters*, and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have a matching name.

- *Version*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have a matching version number.

- *System*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have a matching System value.

- *Type*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have a matching Type value.

- *Dependencies*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have matching dependencies.

- *Description*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have matching descriptions.

- *Python Version*. You can enter text to match, including *special characters* and the ScienceLogic Library Manager page will display only ScienceLogic Libraries that have matching versions.

- *Edited By*. You can enter text to match, including *special characters* and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that were created or last edited by a matching username.

- *Last Edited*. You can select from a list of time periods, and the **ScienceLogic Library Manager** page will display only ScienceLogic Libraries that have been created or edited within that time period.

Special Characters

You can include the following special characters to filter by each column except those that display date and time:

> **NOTE:** When searching for a string, SL1 will match substrings by default, even if you do not include any special characters. For example, searching for "hel" will match both "hello" and "helicopter". When searching for a numeric value, SL1 will not match a substring unless you use a special character.

String and Numeric

- , (comma). Specifies an "OR" operation. Works for string and numeric values. For example:

  "dell, micro" matches all values that contain the string "dell" OR the string "micro".

- & (ampersand). Specifies an "AND " operation. Works for string and numeric values. For example:

    "dell & micro" matches all values that contain both the string "dell" AND the string "micro", in any order.

- ! (exclamation point). Specifies a "not" operation. Works for string and numeric values. For example:

> **NOTE**: You can also use the "!" character in combination with the arithmetical special characters (min-max, >, <, >=, <=, =) described below.

- * (asterisk). Specifies a "match zero or more" operation. Works for string and numeric values. For a string, matches any string that matches the text before and after the asterisk. For a number, matches any number that contains the text. For example:

    "hel*er" would match "helpers" and "helicopter" but not "hello".

    "325*" would match "325", "32561", and "325000".

    "*000" would match "1000", "25000", and "10500000".

- ? (question mark). Specifies "match any one character". Works for string and numeric values. For example:

    "l?ver" would match the strings "oliver", "levers", and "lover", but not "believer".

    "135?" would match the numbers "1350", "1354", and "1359", but not "135" or "13502"

String

- ^ (caret). For strings only. Specifies "match the beginning". Matches any string that begins with the specified string. For example:

    "^sci" would match "scientific" and "sciencelogic", but not "conscious".

    "^happy$" would match only the string "happy", with no characters before or after.

    "!^micro" would match all values that do not start with "micro".

    "!^$" would match all values that are not null.

    "!^" would match null values.

- $ (dollar sign). For strings only. Specifies "match the ending". Matches any string that ends with the specified string. For example:

  "ter$" would match the string "renter" but not the string "terrific".

  "^happy$" would match only the string "happy", with no characters before or after.

  "!fer$" would match all values that do not end with "fer".

  "!^$" would match all values that are not null.

  "!$" would match null values.

> **NOTE**: You can use both ^ and $ if you want to match an entire string and only that string. For example, "^tern$" would match the strings "tern" or "Tern" or "TERN"; it would not match the strings "terne" or "cistern".

Numeric

- min-max. Matches numeric values only. Specifies any value between the minimum value and the maximum value, including the minimum and the maximum. For example:

  "1-5 "would match 1, 2, 3, 4, and 5.

- - (dash). Matches numeric values only. A "half open" range. Specifies values including the minimum and greater or including the maximum and lesser. For example:

  "1-" matches 1 and greater. So would match 1, 2, 6, 345, etc.

  "-5" matches 5 and less. So would match 5, 3, 1, 0, etc.

- \> (greater than). Matches numeric values only. Specifies any value "greater than". For example:

  ">7" would match all values greater than 7.

- < (less than). Matches numeric values only. Specifies any value "less than". For example:

  "<12" would match all values less than 12.

- >= (greater than or equal to). Matches numeric values only. Specifies any value "greater than or equal to". For example:

  "=>7" would match all values 7 and greater.

- <= (less than or equal to). Matches numeric values only. Specifies any value "less than or equal to". For example:

  "=<12" would match all values 12 and less.

- = (equal). Matches numeric values only. For numeric values, allows you to match a negative value. For example:

  "=-5 " would match "-5" instead of being evaluated as the "half open range" as described above.

Examples

- "!dell" matches all values that do not contain the string "dell".
- "!^micro" would match all values that do not start with "micro".
- "!fer$" would match all values that do not end with "fer".
- "!^$" would match all values that are not null.
- "!^" would match null values.
- "!$" would match null values.
- "!*" would match null values.
- "happy, !dell" would match values that contain "happy" OR values that do not contain "dell".
- "aio$". Matches only text that ends with "aio".
- "^shu". Matches only text that begins with "shu".
- "^silo$". Matches only the text "silo", with no characters before or after.
- "!silo". Matches only text that does not contains the characters "silo".
- "!^silo". Matches only text that does not start with "silo".
- "!0$". Matches only text that does not end with "0".
- "!^silo$". Matches only text that is not the exact text "silo", with no characters before or after.
- "!^". Matches null values, typically represented as "--" in most pages.
- "!$". Matches null values, typically represented as "--" in most pages.
- "!^$". Matches all text that is not null.
- silo, !aggr". Matches text that contains the characters "silo" and also text that does not contain "aggr".
- "silo, 02, !aggr". Matches text that contains "silo" and also text that contains "02" and also text that does not contain "aggr".
- "silo, 02, !aggr, !01". Matches text that contains "silo" and also text that contains "02" and also text that does not contain "aggr" and also text that does not contain "01".
- "^s*i*l*o$". Matches text that contains the letter "s", "i", "l", "o", in that order. Other letters might lie between these letters. For example "sXiXlXo" would match.
- "!^s*i*l*o$". Matches all text that does not that contains the letter "s", "i", "l", "o", in that order. Other letters might lie between these letters. For example "sXiXlXo" would not match.
- "!vol&!silo". Matches text that does not contain "vol" AND also does not contain "silo". For example, "volume" would match, because it contains "vol" but not "silo".
- "!vol&02". Matches text that does not contain "vol" AND also contains "02". For example, "happy02" would match, because it does not contain "vol' and it does contain "02".

- "aggr,!vol&02". Matches text that contains "aggr" OR text that does not contain "vol" AND also contains "02".

- "aggr,!vol&!infra". Matches text that contains "aggr" OR text that does not contain "vol" AND does not contain "infra".

- "*". Matches all text.

- "!*". Matches null values, typically represented as "--" in most pages.

- "silo". Matches text that contains "silo".

- " !silo ". Matches text that does not contain "silo".

- " !^silo$ ". Matches all text except the text "silo", with no characters before or after.

- "-3,7-8,11,24,50-". Matches numbers 1, 2, 3, 7, 8, 11, 24, 50, and all numbers greater than 50.

- "-3,7-8,11,24,50-,a". Matches numbers 1, 2, 3, 7, 8, 11, 24, 50, and all numbers greater than 50, and text that includes "a".

- "?n". Matches text that contains any single character and the character "n". For example, this string would match "an", "bn", "cn", "1n", and "2n".

- "n*SAN". Matches text the contains "n", zero or any number of any characters and then "SAN". For example, the string would match "nSAN", and "nhamburgerSAN".

- "^?n*SAN$". Matches text that begins with any single character, is following by "n", and then zero or any number of any characters, and ends in "SAN".

Downloading a ScienceLogic Library

From the **ScienceLogic Library Manager** page, you can download ScienceLogic Libraries. Libraries that are included in SL1 are bundled in either a py_package or a py_directory format.

> **WARNING:** ScienceLogic-authored libraries are copyrighted content and should not be modified without the express permission of ScienceLogic.

To download a ScienceLogic Library:

1. Go to the **ScienceLogic Library Manager** page (System > Customize > ScienceLogic Libraries).
2. Click the save icon ( ) of the ScienceLogic Library you want to download.
3. The library file is downloaded to your computer.

Importing a ScienceLogic Library

In addition to the ScienceLogic Libraries that are included in the latest version of SL1 or included in a PowerPack, you can use the **ScienceLogic Library Manager** page to import additional libraries.

To import a library:

1. Go to the **ScienceLogic Library Manager** page (System > Customize > ScienceLogic Libraries).

2. Click the **[Actions]** menu and then select *Import ScienceLogic Library*.

3. The **Import ScienceLogic Library** dialog box appears. Use the **[Browse]** button to navigate to the ScienceLogic Library file you want to import and then click the **[Import]** button.

4. The imported ScienceLogic Library appears on the **ScienceLogic Library Manager** page.

Creating a ScienceLogic Library

You can create your own library to import into SL1 in either a py_directory or py_package library file saved in uncompressed **.tar** format. This section describes the requirements for a ScienceLogic-compatible library.

Important Notes on Creating ScienceLogic Libraries

The following are important items to note when creating a ScienceLogic library:

- Library files must be saved in uncompressed **.tar** format.

- For best results, use GNU tar rather than bsdtar.

- For a py_directory library, the library name comes from the name of the directory that you saved.

- You can only include wheel files (.whl) as additional dependencies. You cannot include any additional packages in your library file that require compilation or linking because Python build tools are not included in the SL1 platform.

- Any libraries that are bundled as dependencies in your py_package file will not be used if a newer version of that library exists on the appliance *unless* you have a restriction in your package specifying a given version to use.

- For example, if your library "my-library" requires the Python community package "requests" version 2.25.1, you would create the library "requests" version "2.25.1" first. You would then import the requests library into SL1 and align it and "my-library" to your Execution Environment. At runtime, both libraries, at those specific versions, will be used.

- For an alternative example, if your library "my-library" requires the Python community package "requests", but the version is not important, you would include the "requests" package as a ".whl" file in your "my-library" library, adding it to the "manifest.txt" file. After aligning "my-library" to your Execution Environment, it will be available at runtime. The "requests" package included in the "manifest.txt" will be available during the install of "my-library", but may not be used if a newer acceptable package is found for install.

- For a py_package library, an optional "metadata.json" file can be included but it must include the requires_python element, otherwise an error will be displayed upon import.

> **TIP:** A best practice is to version your libraries. Each time you make modifications to a library, create a new version and a new execution environment that uses that version. In that way, you can use different versions in different Dynamic Applications for testing purposes.

Library File Type: py_package

The py_package file type is the preferred library file type used to bundle Python packages and metadata files. A py_package must have the metadata files described below along with the .**whl** file(s), and it must be saved in uncompressed .**tar** format. You must create this tar file with Red Hat Enterprise Linux or a similar operating system to ensure compatibility. Tar files created on MacOS are not supported.

> **NOTE:** All metadata files included in the package must be at the root level of the directory.

Alternately, you can include a "metadata.json" file instead of the following .**txt** files, but the "metadata.json" file must include the requires_python element, otherwise an error will be displayed upon import.

A py_package tar file must include the following files:

- *__init__.py*. This file is required for the directory to be used as a Python package. It identifies this as a package.
- *name.txt*. This text file specifies the name of the library and must match the name of the package being installed.
- *version.txt*. This text file specifies the version of the library and must match the version of the package being installed.
- *type.txt*. This text file specifies that the library is a py_package type library.
- *manifest.txt*. This text file includes the Python package name and version (as specified in the name.txt and version.txt files), followed by a list of additional supporting files that must be deployed to the internal package repository to use the library. For example, if the library is dependent on another package, that other package must be listed in the manifest file.

> **NOTE:** The package name of the library must be the first entry in the manifest file.

- *Wheel (.whl) file*. This is a standard wheel file that you create. See Python documentation for more information about creating wheel files. Wheel files must be Python 2 or Python 2/3 compatible and must be created in Red Hat Enterprise Linux or a similar operating system. You may include multiple wheel files in the py_package.

> **IMPORTANT:** The `tar` command requires that you include all of the files, but not the current working directory. If you include the current working directory, then the import feature cannot find the root files. For the best results, use the following command:
> ```
> tar cvf thing.tar *
> ```
>
> instead of this command:
> ```
> tar cvf thing.tar .
> ```

Library File Type: `py_directory`

The py_directory file type is used to share short code snippets, for example, between Dynamic Applications or SL1 stacks. Files must be organized in a directory. You must then save the directory in uncompressed **.tar** format in Red Hat Enterprise Linux or a similar operating system. Tar files created in MacOS are not supported.

> **NOTE:** The name of the library comes from the name of the directory.

A py_directory tar file can include the following files:

- *__init__.py*. This file is required for the directory to be imported as a Python package.
- *version.txt*. This text file contains a version string specifying the version of the library.
- *dependencies.txt*. This text file includes a comma-separated list of other libraries on which the library is dependent.
- *description.txt*. This text file includes a single-line description of the library's intended use.
- *Python bytecode (.pyc) files or other source files*. These are the additional relevant files that you might want to include in the library.

> **IMPORTANT:** The `tar` command requires that you include all of the files, but not the current working directory. If you include the current working directory, then the import feature cannot find the root files. For the best results, use the following command:
> ```
> tar cvf thing.tar *
> ```
>
> instead of this command:
> ```
> tar cvf thing.tar .
> ```

Manually Building a ScienceLogic Library

To manually build a ScienceLogic library, follow this example:

1. Use the sample python library called `silo_demo-0.0.7-py2.py3-none-any.whl`:

   - Create a project folder for your library, for example, `myproject`.
   - Copy your project's python wheel file into your new project folder.
   ```
   mkdir my project
   ```
   ```
   cp silo_demo-0.0.7-py2.py3-none-any.whl myproject/
   ```

2. Download a complete set of python dependencies for your python package. In this example case there is a dependency on `requests`. To download the dependency package, you need to change the project directory to download the package and any transitive dependencies using the python binary path in your execution environment.

```
cd myproject
```

```
python3 -m pip download --find-links=. silo_demo==0.0.7
```

> **NOTE:** To build these packages, ScienceLogic recommends that you are on Oracle Linux or Redhat based x86-64 installation.

> **NOTE:** Execute the `python3 -m pip download command` with a *python3.6* interpreter to operate correctly with version 11.3 or later of SL1. Some Linux distributions default the command python to *python2.7*.

- The corresponding request python package and all its dependencies will download into your project folder.

```
Looking in links: .

Processing ./silo_demo-0.0.7-py2.py3-none-any.whl

    File was already downloaded /ScienceLogic/myproject/silo_
    demo-0.0.7-py2.py3-none-any.whl

Collecting requests

    Using cached requests-2.29.0-py3-none-any.whl (62 kB)

Collecting certifi>=2017.4.17

    Using cached certifi-2022.12.7-py3-none-any.whl (155 kB)

Collecting urllib3<1.27,>=1.21.1

    Using cached urllib3-1.26.15-py2.py3-none-any.whl (140 kB)

Collecting charset-normalizer<4,>=2

    Using cached charset_normalizer-3.1.0-cp39-cp39-macosx_10_
    9_x86_64.whl (124 kB)

Collecting idna<4,>=2.5

    Using cached idna-3.4-py3-none-any.whl (61 kB)

Saved ./requests-2.29.0-py3-none-any.whl

Saved ./certifi-2022.12.7-py3-none-any.whl

Saved ./charset_normalizer-3.1.0-cp39-cp39-macosx_10_9_x86_
64.whl

Saved ./idna-3.4-py3-none-any.whl

Saved ./urllib3-1.26.15-py2.py3-none-any.whl

Successfully downloaded silo_demo requests certifi charset-
normalizer idna urllib3
```

- List the dependencies to ensure they are correct.

```
[myproject]$ ls -l

-rw-r--r--@ 1 dev staff 4524 May 2 11:31 silo_demo-0.0.7-
py2.py3-none-any.whl

-rw-r--r-- 1 dev staff 62499 May 3 16:31 requests-2.29.0-py3-
none-any.whl

-rw-r--r-- 1 dev staff 155255 May 3 16:31 certifi-2022.12.7-
py3-none-any.whl

-rw-r--r-- 1 dev staff 124711 May 3 16:31 charset_normalizer-
2.0.4-cp36-cp36mu-manylinux1_x86_64.whl

-rw-r--r-- 1 dev staff 61538 May 3 16:31 idna-3.4-py3-none-
any.whl

-rw-r--r-- 1 dev staff 140881 May 3 16:31 urllib3-1.26.15-
py2.py3-none-any.whl
```

> **CAUTION:** SL1 comes pre-installed with the proper versions of wheel, setuptools, and pip wheels for each release. Do not include these in your ScienceLogic library as it will cause a deployment failure.

3. Create the *metadata.json* file. Ensure that you are in your project folder and use your favorite editor tool to create the *metadata.json* file.

   The following example includes the minimum set of metadata required by SL1. The information supplied corresponds to the principle python library you set up in step 1. The environment deployer will `pip install` the principle python library based on the `name` and `version` metadata, so they must be accurate.

   Update the following fields with your information:
   - ***meta_version***. Leave this as default `"0.2"`
   - ***description***. A precise description of the ScienceLogic library
   - **manifest**. Contains the complete list of wheel files from Step 2
   - ***name***. The precise name of the principle python package used by pip to install the python package. In this example, the name is `silo_demo`.
   - ***type***. Leave this as default `"py_package"`
   - ***version***. The precise version of the principle python package used by pip to install the python package. In this example, the version is `0.0.7`.
   - ***requires_python*** The specific versions of python you will use. For python2, use `"~=2.7"` and for python3, use `"~=3.6"`. You should follow PEP440 version specifiers.

> **NOTE:** For more information about the PEP440 version specifiers, see the [PEP440 webpage.](#)

```json
{

    "meta_version": "0.2",

    "fields": {

            "description": "A demo ScienceLogic library",

            "manifest": ["certifi-2022.12.7-py3-none-any.whl",
            "charset_normalizer-2.0.4-cp36-cp36mu-manylinux1_x86_
            64.whl", "idna-3.4-py3-none-any.whl", "requests-2.29.0-
            py3-none-any.whl", "silo_demo-0.0.7-py2.py3-none-any.whl",
            "urllib3-1.26.15-py2.py3-none-any.whl"],

            "name": "silo_demo",

            "type": "py_package",

            "version": "0.0.7",

            "requires_python": "~=3.6"

    }

}
```

4. Validate the *metadata.json* file. Validating your JSON files ensures there aren't any errors in the pairing of braces and syntax. To validate your metadata.json file type:

```
python -m json.tool metadata.json
```

5. Create the ScienceLogic library archive.

Give your archive a unique name so that it is easy to recognize and distinguish from other library archives. Since it is possible to support multiple python runtime versions, ScienceLogic recommends that you include the python version(s) supported in the library archive you create.

Use the following naming convention as a guide:

```
<lib name> + <lib version>. <runtime version/s>.tar
```

For example:

```
tar cvf silo_demo+0.0.7.py36.tar *
```

```
ls -l *.tar
```

```
-rw-r--r-- 1 em7admin em7admin 480256 Nov 29 16:28 silo_
demo+0.0.7.py36.tar
```

```
tar tvf silo_demo+0.0.7.py36.tar
```

```
-rw-r--r-- 1 em7admin em7admin 161120 Nov 29 16:12 certifi-2022.9.24-py3-
none-any.whl
```

```
-rw-r--r-- 1 em7admin em7admin 39748 Nov 29 16:12 charset_normalizer-
2.0.4-cp36-cp36mu-manylinux1_x86_64.whl
```

```
-rw-r--r-- 1 em7admin em7admin 61538 Nov 29 16:12 idna-3.4-py3-none-
any.whl
```

```
-rw-r--r-- 1 em7admin em7admin 479 Nov 29 16:25 metadata.json
```

```
-rw-r--r-- 1 em7admin em7admin 62843 Nov 29 16:12 requests-2.28.1-py3-
none-any.whl
```

```
-rw-r--r-- 1 em7admin em7admin 4524 Nov 29 16:12 silo_demo-0.0.7-py2.py3-
none-any.whl
```

```
-rw-r--r-- 1 em7admin em7admin 140572 Nov 29 16:12 urllib3-1.26.13-
py2.py3-none-any.whl
```

> CAUTION: Depending on your development environment, your .tar archiving tool may generate an archive that is incompatible with the Linux distribution used by SL1. If this occurs, try extracting (`tar xvf myarchive.tar`) or just listing the archive (`tar tvf my archive.tar`) on your SL1 appliance before archiving to ensure compatibility with the following steps.

6. Import the ScienceLogic library.

To import a ScienceLogic library, see *Importing a ScienceLogic Library*.

7. Create an SL1 Execution Environment.
   - To create your python3 Execution Environment, go to the Environment Manager page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments) and click **[New]**.
   - In the **Environment Name** field, select *python3*.
   - For this example, add the **Environment Name** `Snippet Framework Demo py36` and click **[Save]**. The compatible libraries are displayed and available to the python3 execution environment.
   - Click the lightning bolt (🖊) to include all of the libraries in this environment.

> **IMPORTANT:** The string found in `requires_python` will follow the same convention used for python packages.

> **NOTE:** A ScienceLogic library with `requires_python>=3.6` would be compatible with any execution environment greater than or equal to 3.6.

- Make note of the environment GUID for the following steps. In this example, the environment GUID was:

  ```
  0117E1C041D8CE0C69A8391C02E6DC3B
  ```

8. Align your ScienceLogic library to the SL1 Execution Environment.

   - Return to the **Environment Editor** page (System>Customize>ScienceLogic Libraries>Actions>Execution Environments). You should see that your ScienceLogic library is available for alignment to your execution environment.
   - Click the lightning bolt ( ) to add your library to the environment.

9. Test your ScienceLogic library.

   - Log on to the collector and deploy the ScienceLogic Execution Environment.

   ```
   sudo -u s-em7-core /opt/em7/bin/python3 /opt/em7/lib/python3/sl_envs/env_
   manager.pyc --debug 0117E1C041D8CE0C69A8391C02E6DC3B
   ```

   - Activate your new environment.

   ```
   source /opt/em7/envs/0117E1C041D8CE0C69A8391C02E6DC3B/bin/activate
   ```

   - From the python command line, try importing your new package and executing its API.

   ```
   [uenv-3869008638788123352]$ python

   Python 3.6.8 (default, Nov 18 2021, 10:07:16)

   [GCC 4.8.5 20150623 (Red Hat 4.8.5-44.0.3)] on linux

   Type "help", "copyright", "credits" or "license" for more information.

   >>> from silo.demo import demo_class

   >>>
   ```

> **NOTE:** ScienceLogic is working to minimize the development effort through the use of Snippet Framework and low code specifications. Unless your collection environment is unique, you may not need to develop content such as manually building your ScienceLogic library.

Deleting a ScienceLogic Library

The **ScienceLogic Library Manager** page allows you to delete ScienceLogic Libraries that are not currently in use.

SL1 will not allow you to delete ScienceLogic Libraries that are aligned to execution environments that are being used by any Dynamic Applications, credential tests, or Run Book Actions. You also cannot delete libraries that are aligned to the "System" environment.

To delete a library:

1. Go to the **ScienceLogic Library Manager** page (System > Customize > ScienceLogic Libraries).
2. Select the ScienceLogic Library that you want to delete.
3. Click the ***Select Actions*** menu in the lower right of the page and select *DELETE ScienceLogic Library*, and then click the **[Go]** button.
4. The selected ScienceLogic Library is deleted from the **ScienceLogic Library Manager** page.

Chapter

3

Managing Execution Environments

Overview

Execution environments let you share code between snippets, Dynamic Applications, and PowerPacks. Execution environments have a global unique identifier (GUID), a name, and list of associated libraries. The libraries associated with an execution environment provide a way to bundle reusable code in a self-contained environment available in snippets. You can use different execution environments for different Dynamic Application requirements.

This chapter covers the following topics:

Viewing the List of Execution Environments

The **Environment Manager** page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments) displays a list of all existing execution environments.
For each execution environment, the page displays the following information:

> **TIP:** To sort the list of execution environments, click on a column heading. The list will be sorted by the
> column value, in ascending order. To sort by descending order, click the column heading again.

- **Environment Name**. The name of the execution environment.
- **Env GUID**. The execution environment's globally unique identifier.
- **Env Type**. The Python runtime version selected for the execution environment.
- **Libs**. The number of ScienceLogic Libraries aligned with the execution environment. Click the View Aligned Libraries icon ( ) to view the list of ScienceLogic Libraries that are aligned to a particular execution environment.

Searching and Filtering the List of Execution Environments

The Filter-While-You-Type fields appear as a row of blank fields at the top of the list. These fields let you filter the items that appear in the list.

The list is dynamically updated as you select each filter. For each filter, you must make a selection from a drop-down menu or type text to match against. SL1 will search for entries that match the text, including partial matches. Text matches are not case-sensitive, and you can use special characters in each text field.

By default, the cursor is placed in the first Filter-While-You-Type field. You can use the <Tab> key or your mouse to move your cursor through the fields.

You can filter by one or more of the following parameters. Only items that meet all of the filter criteria are displayed on the page.

- **Environment Name**. You can enter text to match, including *special characters*, and the **Environment Manager** page will display only execution environments that have a matching name.
- **Env GUID**. You can enter text to match, including *special characters*, and the **Environment Manager** page will display only execution environments that have a matching GUID.
- **Env Type**. You can enter text to match, and the **Environment Manager** page will display only execution environments that have a matching Python runtime version for the execution environment.
- **Libs**. You can enter text to match, including *special characters*, and the **Environment Manager** page will display only execution environments that have a matching number of aligned ScienceLogic Libraries.

Special Characters

You can include the following special characters to filter by each column except those that display date and time:

> **NOTE:** When searching for a string, SL1 will match substrings by default, even if you do not include any special characters. For example, searching for "hel" will match both "hello" and "helicopter". When searching for a numeric value, SL1 will not match a substring unless you use a special character.

String and Numeric

- **, (comma).** Specifies an "OR" operation. Works for string and numeric values. For example:

  "dell, micro" matches all values that contain the string "dell" OR the string "micro".

- **& (ampersand).** Specifies an "AND " operation. Works for string and numeric values. For example:

  "dell & micro" matches all values that contain both the string "dell" AND the string "micro", in any order.

- **! (exclamation point).** Specifies a "not" operation. Works for string and numeric values. For example:

> **NOTE**: You can also use the "!" character in combination with the arithmetical special characters (min-max, >, <, >=, <=, =) described below.

- *** (asterisk).** Specifies a "match zero or more" operation. Works for string and numeric values. For a string, matches any string that matches the text before and after the asterisk. For a number, matches any number that contains the text. For example:

  "hel*er" would match "helpers" and "helicopter" but not "hello".

  "325*" would match "325", "32561", and "325000".

  "*000" would match "1000", "25000", and "10500000".

- **? (question mark).** Specifies "match any one character". Works for string and numeric values. For example:

  "l?ver" would match the strings "oliver", "levers", and "lover", but not "believer".

  "135?" would match the numbers "1350", "1354", and "1359", but not "135" or "13502"

String

- ^ (caret). For strings only. Specifies "match the beginning". Matches any string that begins with the specified string. For example:

  " ^ sci" would match "scientific" and "sciencelogic", but not "conscious".

  " ^ happy$" would match only the string "happy", with no characters before or after.

  "! ^ micro" would match all values that do not start with "micro".

  "! ^ $" would match all values that are not null.

  "! ^ " would match null values.

- $ (dollar sign). For strings only. Specifies "match the ending". Matches any string that ends with the specified string. For example:

  "ter$" would match the string "renter" but not the string "terrific".

  " ^ happy$" would match only the string "happy", with no characters before or after.

  "!fer$" would match all values that do not end with "fer".

  "! ^ $" would match all values that are not null.

  "!$" would match null values.

NOTE: You can use both ^ and $ if you want to match an entire string and only that string. For example, " ^ tern$" would match the strings "tern" or "Tern" or "TERN"; it would not match the strings "terne" or "cistern".

Numeric

- min-max. Matches numeric values only. Specifies any value between the minimum value and the maximum value, including the minimum and the maximum. For example:

  "1-5 "would match 1, 2, 3, 4, and 5.

- - (dash). Matches numeric values only. A "half open" range. Specifies values including the minimum and greater or including the maximum and lesser. For example:

  "1-" matches 1 and greater. So would match 1, 2, 6, 345, etc.

  "-5" matches 5 and less. So would match 5, 3, 1, 0, etc.

- > (greater than). Matches numeric values only. Specifies any value "greater than". For example:

  ">7" would match all values greater than 7.

Viewing the List of Execution Environments

- < (less than). Matches numeric values only. Specifies any value "less than". For example:

  "<12" would match all values less than 12.

- >= (greater than or equal to). Matches numeric values only. Specifies any value "greater than or equal to". For example:

  "=>7" would match all values 7 and greater.

- <= (less than or equal to). Matches numeric values only. Specifies any value "less than or equal to". For example:

  "=<12" would match all values 12 and less.

- = (equal). Matches numeric values only. For numeric values, allows you to match a negative value. For example:

  "=-5 " would match "-5" instead of being evaluated as the "half open range" as described above.

Examples

- "!dell" matches all values that do not contain the string "dell".
- "!^micro" would match all values that do not start with "micro".
- "!fer$" would match all values that do not end with "fer".
- "!^$" would match all values that are not null.
- "!^" would match null values.
- "!$" would match null values.
- "!*" would match null values.
- "happy, !dell" would match values that contain "happy" OR values that do not contain "dell".
- "aio$". Matches only text that ends with "aio".
- "^shu". Matches only text that begins with "shu".
- "^silo$". Matches only the text "silo", with no characters before or after.
- "!silo". Matches only text that does not contains the characters "silo".
- "!^silo". Matches only text that does not start with "silo".
- "!0$". Matches only text that does not end with "0".
- "!^silo$". Matches only text that is not the exact text "silo", with no characters before or after.
- "!^". Matches null values, typically represented as "--" in most pages.
- "!$". Matches null values, typically represented as "--" in most pages.
- "!^$". Matches all text that is not null.
- silo, !aggr". Matches text that contains the characters "silo" and also text that does not contain "aggr".
- "silo, 02, !aggr". Matches text that contains "silo" and also text that contains "02" and also text that does not contain "aggr".

- "silo, 02, !aggr, !01". Matches text that contains "silo" and also text that contains "02" and also text that does not contain "aggr" and also text that does not contain "01".

- " ^ s*i*l*o$". Matches text that contains the letter "s", "i", "l", "o", in that order. Other letters might lie between these letters. For example "sXiXlXo" would match.

- "! ^ s*i*l*o$". Matches all text that does not that contains the letter "s", "i", "l", "o", in that order. Other letters might lie between these letters. For example "sXiXlXo" would not match.

- "!vol&!silo". Matches text that does not contain "vol" AND also does not contain "silo". For example, "volume" would match, because it contains "vol" but not "silo".

- "!vol&02". Matches text that does not contain "vol" AND also contains "02". For example, "happy02" would match, because it does not contain "vol' and it does contain "02".

- "aggr,!vol&02". Matches text that contains "aggr" OR text that does not contain "vol" AND also contains "02".

- "aggr,!vol&!infra". Matches text that contains "aggr" OR text that does not contain "vol" AND does not contain "infra".

- "*". Matches all text.

- "!*". Matches null values, typically represented as "--" in most pages.

- "silo". Matches text that contains "silo".

- " !silo ". Matches text that does not contain "silo".

- " ! ^ silo$ ". Matches all text except the text "silo", with no characters before or after.

- "-3,7-8,11,24,50-". Matches numbers 1, 2, 3, 7, 8, 11, 24, 50, and all numbers greater than 50.

- "-3,7-8,11,24,50-,a". Matches numbers 1, 2, 3, 7, 8, 11, 24, 50, and all numbers greater than 50, and text that includes "a".

- "?n". Matches text that contains any single character and the character "n". For example, this string would match "an", "bn", "cn", "1n", and "2n".

- "n*SAN". Matches text the contains "n", zero or any number of any characters and then "SAN". For example, the string would match "nSAN", and "nhamburgerSAN".

- " ^ ?n*SAN$". Matches text that begins with any single character, is following by "n", and then zero or any number of any characters, and ends in "SAN".

Creating an Execution Environment

SL1 includes several execution environments from preloaded PowerPacks and system updates for your use. If needed, you can also create new execution environments. After you have successfully created an execution environment, it will appear in the list of execution environments available for aligning with Dynamic Applications, credential tests, and Run Book Actions.

To create an execution environment:

1. From the **Environment Manager** page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments), click the **[New]** button.

2. On the **Environment Editor** page, type a name for the new execution environment in the *Environment Name* field, select a Python version from the *Environment Type* drop-down, and then click the save icon (![save icon]).

> NOTE: Each execution environment is assigned an Env GUID, which makes it unique. Since two environments can share a name, it is a best practice to use the Name field to version an environment when updating its contents.

3. The **Library Alignment** pane appears, displaying a list of ScienceLogic Libraries that you can align with the new execution environment. Click the align icon (![align icon]) for a ScienceLogic Library to align that library to the new execution environment.

> TIP: You can select the *Filter Libraries By Python Version* drop-down to filter ScienceLogic libraries compatible with the execution environment runtime. This feature is set to *on* by default.

4. Align any additional ScienceLogic libraries to the execution environment as needed.

> TIP: If necessary, you can click the unalign icon (![unalign icon]) for a ScienceLogic Library to remove that library's alignment with the new execution environment.

5. When you are finished, click the save icon (![save icon]) again to save the execution environment with the ScienceLogic Library alignment you selected, and then click **[Close]**.

Editing an Execution Environment

You can edit an execution environment by clicking its wrench icon (![wrench icon]). When you do so, you can change the environment's name, align additional ScienceLogic Libraries to the environment, or remove its alignment with one or more libraries.

> WARNING: Do not edit ScienceLogic-created execution environments. You may make a copy of the execution environment, but ScienceLogic often updates these execution environments in a release, which will overwrite any existing execution environment of the same globally unique identifier (GUID).

To edit an execution Environment:

1. From the **Environment Manager** page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments), click the edit icon (![edit icon]) for the execution environment that you want to edit.

2. On the **Environment Editor** page, you can do any of the following:

- Change the environment's name in the **Environment Name** field.

- Align additional ScienceLogic Libraries to the execution environment by clicking the appropriate align icons ( ).

- Remove the execution environment's alignment with one or more libraries by clicking the appropriate unalign icons ( ).

3. When you are finished, click the save icon ( ) and then click **[Close]**.

Copying an Execution Environment

If you want to create a new execution environment that has traits similar to an existing environment, you can make a copy of the existing execution environment that you can then edit as needed. When you copy an existing execution environment, the new environment includes the same libraries as the original environment, but it will have a new globally unique identifier (GUID).

> **CAUTION:** Making a copy of an execution environment does not mean there will be a duplicate runtime environment. Environments that share a set of libraries will result in the same virtual runtime environment to optimize space on the Data Collector.

To copy an execution environment:

1. From the **Environment Manager** page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments), click the copy icon ( ) for the execution environment that you want to copy. A confirmation message appears.

2. Click **[OK]** to continue. A copy of the selected execution environment appears on the **Environment Manager** page. By default, it has the same name as the original execution environment, followed by the word "(copy)".

Deleting an Execution Environment

If you no longer need an execution environment and it is not currently aligned with a Dynamic Application, credential test, or Run Book Action, then you can delete it. SL1 will not allow you to delete execution environments that are being used by any Dynamic Applications, credential tests, or Run Book Actions.

> **CAUTION:** Do not delete execution environments that are shipped with SL1.

To delete an execution environment:

1. From the **Environment Manager** page (System > Customize > ScienceLogic Libraries > Actions > Execution Environments), click the delete icon ( ) for the execution environment that you want to delete.

2. A confirmation message appears. Click **[OK]** to continue.

3. The execution environment is deleted from the **Environment Manager** page.