How to Create and Share SPASE Resource Descriptions

2021-Mar-01

Table of Contents

Introduction 1

Concepts 2

Identifiers 2

Tools, Specifications and Conventions 2

- Tools 3
- Specifications 4
- Conventions 4

Creating Descriptions 4

- Preliminary Checks 6
- Repository Organization 6
- XML Special Character 6
- Formatted Text 7
- Making Description Look Good 8

Versioning 8

Validation and Referential Checks 8

- Single Description 8
- Large Collections of Descriptions 9

Sharing 9

- Landing Pages 10

DOI Minting 10

Harvesting 10

Tips and Tricks 11

Final Thoughts 11

Introduction

A SPASE resource description is a set of metadata which captures the attributes and characteristics of a resource. A resource is anything that contributes to the acquisition of data and the data itself. This includes the people involved, the observatory, instruments, documentation, repositories for the data, services, registry of resources. Data can be the original observations, derived or processed. In order to have coherent and easily exchanged resource descriptions the SPASE Information Model is used to classify and structure the metadata.

Concepts

The SPASE Information Model takes a declarative approach to building connections between resources. That is, a resource declares itself to be a part of something larger or to have originated from some other resource. A declarative approach minimizes changes to the resource description for things that do not inherently change. For example, an instrument does not change, but it may generate a large number of data sets. As new data sets are created only the new data set needs to have a SPASE descriptions created and the instrument descriptions does not need to change. For example, suppose you have magnetic field data from the Ulysses spacecraft.

- The data set originates from magnetometer instrument.
- The magnetometer instrument is attached to the Ulysses observatory

The data set SPASE description will indicate with an identifier the instrument it was acquired from. In turn the instrument SPASE description will indicate with an identifier which observatory it was part of.

Identifiers

A resource description is intended to be shared and referenced by other resources. To aid in this, every resource is assigned a unique identifier. A resource identifier is a URI that has the form

> scheme://authority/type/path

where "scheme" is "spase" for those resources administered through the SPASE framework, "authority" is the unique identifier for the naming authority which manages the resource, "type" is the resource type and "path" is the unique local identifier of the resource within the context of the "authority". As long as the naming authority ensures that the "type/path" is unique within their namespace, it will also be unique within the entire data environment.

To illustrate the definition of a resource identifier consider that there is a registered "authority" called "NASA" which maintains information for NASA related spacecraft (Observatory) resources. One such spacecraft is GOES8. Since this is a description for an observatory the resource type is "Observatory". Now "NASA" decides that the "path" to the GOES8 resource description should be the common name for the spacecraft which is "GOES8". So, the resource identifier would be:

> spase://NASA/Observatory/GOES8

Tools, Specifications and Conventions

The SPASE community maintains a web site with links to the tools, specifications and recommendations for using the SPASE Information Model. This web site is located at:

> https://spase-group.org/

Tools

There are two general types of tools. One is on-line tools that can be used in a web browser and the other are command-line tools that can be used on your local system. The on-line tools are:

Editor: http://xmleditor.spase-group.org/

> You can create resources descriptions from scratch or load an existing resource description and make changes. Using the on-line can be very useful when creating single descriptions such as for an observatory, instrument, person or data.

Validator: http://validator.spase-group.org/

> You can validate a single resource description. Validation involves checking that the resource description conforms to the Information Model specification. You can also perform a referential check on URLs and resource IDs in the description. This ensures that a resource description is complete and accurate.

On-line Tools

> The SPASE Resource Tools package contains a set of command-line applications which can be used to generate, validate, referentially check, use and organize resource descriptions written in SPASE XML.
>
> The SPASE Resource Tools is written Node.js and should run on any system with a Node installed.
>
> Installation

```
npm install spase-resource-tools -g
```

> After installation the tools will be available at the command line. Each tool has a "spase-" prefix to distinguish it from any other similar tools. Available tools are:

spase-collate: Separate each SPASE resource description in a file into a separate file stored in a folder tree according to the Resource ID.

spase-refcheck: Check resource identifiers and URLs for referential integrity.

spase-validate: Validate a SPASE resource description using a specified version of the data dictionary (XML schema).

spase-tree: List the file tree. Optionally list only files with a given extension.

spase-update-authority: Change the control authority in the ResourceID, add a PriorID and update ReleaseDate.

spase-doi-ref: Extract reference information in CSV format from a SPASE resource description.

spase-doi-request: Extract information from a SPASE resource description and generate a DataCite formated DOI request that can be submitted through EZID web service API.

spase-list-elem: List an element value in an XML document.

spase-restamp: Update the to the current data and time.

spase-pretty: Make a pretty XML file by formatting the XML file with indentation and wrapping.

spase-transform : Transform a SPASE description in XML using an XML Stylesheet.

Specifications

The SPASE community manages the SPASE Information Model and makes versioned releases. The specification for each version of the Information Model can be found at:

https://spase-group.org/data/model

This instance is referred to as the "Base Model". Extensions to the base model are defined for specialized forms of data. There is a extension for describing simulations, simulation runs and the data output from the runs. This can be found at:

https://spase-group.org/data/simulation/

Conventions

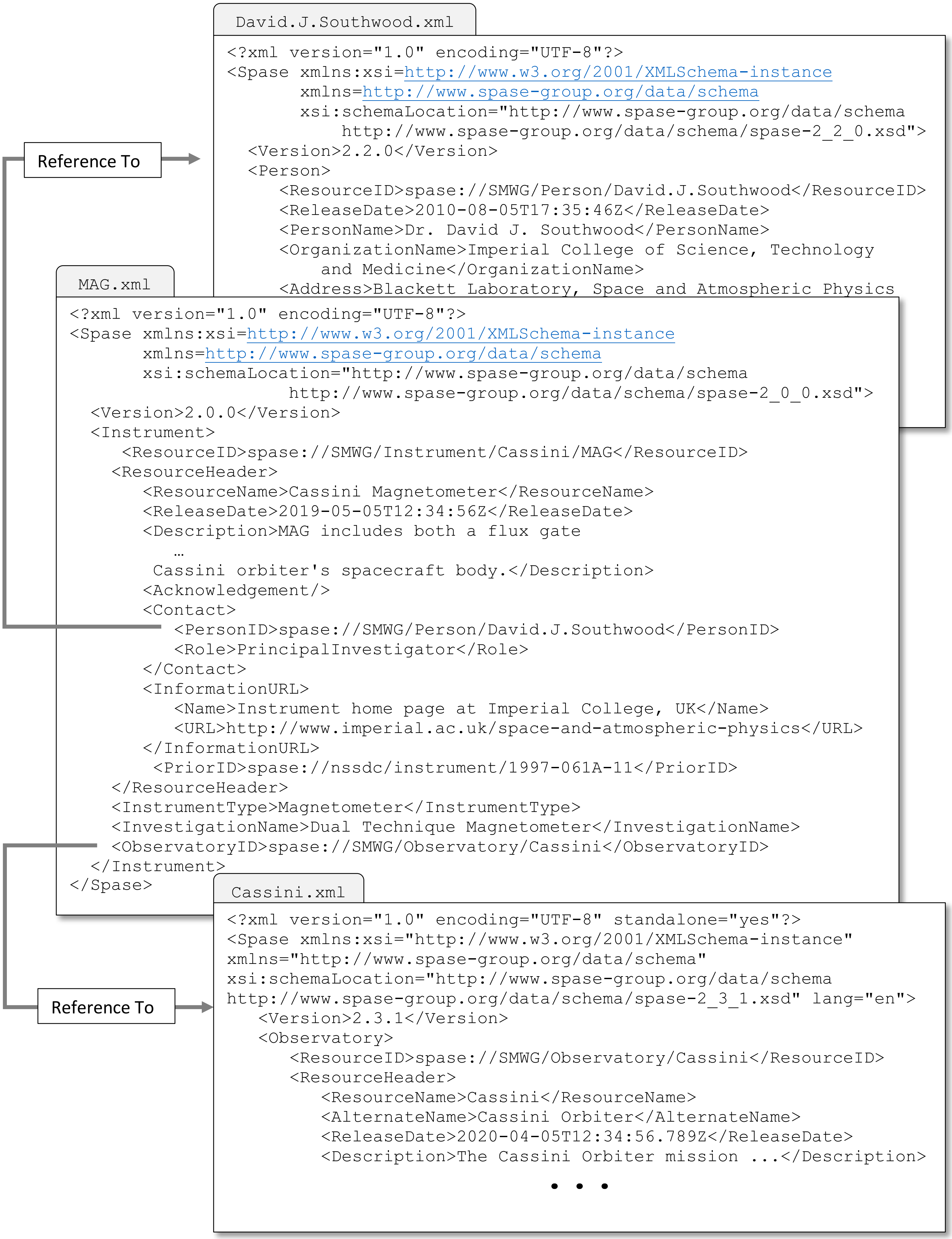
The SPASE community has defined a set of conventions which are best practices for creating quality resource descriptions. The conventions can be found at:

https://spase-group.org/docs/conventions/

Creating Descriptions

A SPASE descriptions is an XML document that conforms to the SPASE Information Model specification. You can create a SPASE resource description with any suitable editor. Some options are to use the SPASE on-line editor (http://xmleditor.spase-group.org/), a plain text editor like Notepad++ (https://notepad-plus-plus.org/) or a commercial XML editor like Oxygen (https://www.oxygenxml.com/). Both the SPASE on-line editor and Oxygen are schema driven editors, that is, an XML schema is used to aid in constructing a conforming XML document. For the SPASE on-line editor the schema is automatically loaded. For Oxygen you will need to down the XML schema from https://spase-group.org/data/model. For Notepad++ adding the "XML Tools" plugin is very useful.

It can be very instructive to look at existing SPASE descriptions to get familiar with the contents of a typical XML document. A good source is the HPDE landing pages located at https://hpde.io At this site you can navigate to a SPASE description similar to what you plan create and then download the XML version. An example is show in Figure 1 which shows the references in a typical Instrument description.



David.J.Southwood.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Spase xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
       xmlns=http://www.spase-group.org/data/schema
       xsi:schemaLocation="http://www.spase-group.org/data/schema
           http://www.spase-group.org/data/schema/spase-2_2_0.xsd">
  <Version>2.2.0</Version>
  <Person>
     <ResourceID>spase://SMWG/Person/David.J.Southwood</ResourceID>
     <ReleaseDate>2010-08-05T17:35:46Z</ReleaseDate>
     <PersonName>Dr. David J. Southwood</PersonName>
     <OrganizationName>Imperial College of Science, Technology
         and Medicine</OrganizationName>
     <Address>Blackett Laboratory, Space and Atmospheric Physics
```

MAG.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Spase xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
       xmlns=http://www.spase-group.org/data/schema
       xsi:schemaLocation="http://www.spase-group.org/data/schema
              http://www.spase-group.org/data/schema/spase-2_0_0.xsd">
  <Version>2.0.0</Version>
  <Instrument>
     <ResourceID>spase://SMWG/Instrument/Cassini/MAG</ResourceID>
    <ResourceHeader>
       <ResourceName>Cassini Magnetometer</ResourceName>
       <ReleaseDate>2019-05-05T12:34:56Z</ReleaseDate>
       <Description>MAG includes both a flux gate
          …
        Cassini orbiter's spacecraft body.</Description>
       <Acknowledgement/>
       <Contact>
          <PersonID>spase://SMWG/Person/David.J.Southwood</PersonID>
          <Role>PrincipalInvestigator</Role>
       </Contact>
       <InformationURL>
          <Name>Instrument home page at Imperial College, UK</Name>
          <URL>http://www.imperial.ac.uk/space-and-atmospheric-physics</URL>
       </InformationURL>
        <PriorID>spase://nssdc/instrument/1997-061A-11</PriorID>
    </ResourceHeader>
    <InstrumentType>Magnetometer</InstrumentType>
    <InvestigationName>Dual Technique Magnetometer</InvestigationName>
    <ObservatoryID>spase://SMWG/Observatory/Cassini</ObservatoryID>
  </Instrument>
</Spase>
```

Cassini.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Spase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.spase-group.org/data/schema"
xsi:schemaLocation="http://www.spase-group.org/data/schema
http://www.spase-group.org/data/schema/spase-2_3_1.xsd" lang="en">
   <Version>2.3.1</Version>
   <Observatory>
      <ResourceID>spase://SMWG/Observatory/Cassini</ResourceID>
      <ResourceHeader>
         <ResourceName>Cassini</ResourceName>
         <AlternateName>Cassini Orbiter</AlternateName>
         <ReleaseDate>2020-04-05T12:34:56.789Z</ReleaseDate>
         <Description>The Cassini Orbiter mission ...</Description>
                             • • •
```

Figure 1: Instrument description with referenced Observatory and Person descriptions. Text in <Description> elements and the Observatory description is abridged for illustration purposes.

Preliminary Checks

Depending on the type of resource you are creating there may be some additional SPASE descriptions you need to create. The reason for this is that a SPASE description may reference a resource description of a different type. References are made using the resource ID associated with the resource, so a resource descriptions needs to exist before it can referenced. One example of using references is a SPASE description of an instrument. An instrument description will typically identify who the Principal Investigator (PI) is for the instrument. This is done by referencing a Person description. You can check at https://hpde.io for the required Person description, but if one does not exist you will need to create one. An instrument description may also reference the observatory it is associated with. An observatory can be a lab, facility, a ground station or spacecraft. Again, you can check at https://hpde.io for the required Person description, but if one does not exist you will need to create one.

Repository Organization

The organization of a repository follows the hierarchy of the ResourceIDs assigned in the SPASE description. This means that a repository has a root folder which matches the naming authority. In this folder are folders with the name that matches the resource type. Depending on the type resource there may be XML files containing SPASE descriptions are there may be folders which match the observatory name. For example, a Person resource type would contain XML files with Person descriptions. A NumericalData resource type would contain folders with observatory names.

The organization of a typical repository looks something like:

```
NASA -- ┌ DisplayData
        ⊦ NumericalData
        ⊦ Instrument
        ⊦  Observatory
        └ Person
```

There is a tool class "spase-collate" in the SPASE Resource Tools package to help organize SPASE descriptions based on the ResourceID. Running "spase-collate" on a SPASE description will place it in the proper folder. For example, suppose you have SPASE descriptions is a folder called "draft" and want to collate and place them the folder "release". Use the following command in parent folder of "draft":

```
spase-collate -b release -r draft
```

Regardless of the organization of the files in "draft", the files in "release" will be organized in a proper hierarchy.

XML Special Character

One XML schema that most developers are familiar with is HTML. SPASE descriptions are also based on an XML schema. There is often a tendency to try to use HTML tags to represent special characters in the text of a SPASE description. This will not work because in XML the characters "<" and "&" are key syntax markers. However special characters can be included in text using XML entities. The XML specification includes five pre-declared entities to represent so-called special characters:

Symbol Name	Symbol	Entity
Ampersand	&	&amp;

Less than	<	&lt;
Greater than	>	&gt;
Apostrophe	'	&apos;
Quote	"	&quot;

Other special characters are available in the UTF-8 character set. To use these characters you need to a numeric character reference which has the format &#nnnn;. Commonly used mathematical characters are:

Symbol Name	Symbol	Entity
Degree symbol	°	&#176;
Plus-minus symbol	±	&#177;
Multiplication symbol	×	&#215;
Tilde operator	∼	&#8764;
Less than or equal to symbol	≤	&#8804;

And commonly used Greek letters are:

Symbol Name	Symbol	Entity
Cedilla	¸	&#184
Angstrom symbol	Å	&#8491;
Latin capital letter A with ring above	Å	&#197;
Latin capital letter O with diaeresis	Ö	&#214;
Latin capital letter O with stroke	Ø	&#216;
Latin small letter e with acute	é	&#233;
Latin small letter e with circumflex	ê	&#234;
Latin small letter n with tilde	ñ	&#241;
Latin small letter o with stroke	ø	&#248;
Latin small letter u with diaeresis	ü	&#252;
Greek capital letter delta	Δ	&#916;
Greek capital letter sigma	Σ	&#931;
Greek capital letter phi	Φ	&#981;
Greek small letter beta	β	&#946;
Greek small letter theta	θ	&#952;
Greek small letter pi	π	&#960;
Greek small letter sigma	σ	&#963;
Greek small letter mu	μ	&#956;
Micro symbol	µ	&#181;

Formatted Text

Even though HTML cannot be used to format text, SPASE does allow for free form text to be written using GitHub Flavored Markdown (GFM) (https://github.github.com/gfm/). This is most commonly used in <Description> tags. Tools that render the contents of a SPASE description should format the text when displayed.

Making Description Look Good

Many XML editors will make your SPASE descriptions look good by indenting elements and wrapping text. Formatting XML documents in this way makes it easier for humans, but is not necessary for computers. If you have generated SPASE descriptions that you'd like to make pretty you can use the tool "spase-pretty". Here are some examples:

Format the XML in "example.xml" and write output to the display:

```
spase-pretty example.xml
```

Format the XML in "example.xml" and write output to original file:

```
spase-pretty -w example.xml
```

Format the all the XML files in the current folder and below:

```
spase-pretty -r .
```

Versioning

In a SPASE description there is a <Version> element. This element is for the version of the information model used for the generation of the SPASE description. This allows for easier validation of the SPASE description. The "version" of the description is indicated by the <ReleaseDate> in <ResourceHeader>. This should the date and time of the last update of the SPASE description. The tool "spase-restamp" can be used to set the <ReleaseDate> to the current date and time. For example,

```
spase-restamp -w example.xml
```

will change the <ReleaseDate> in "example.xml" to the current date and time.

Validation and Referential Checks

Quality assurance is an important part of data environment, especially with a distributed organization like is true for the Heliophysics Data Environment. There are two parts to automated validation. The first is the syntactic validation of checking if a SPASE description conforms to the appropriate version of the SPASE information model. The second is a referential check that validates that all references made to other resources can be resolved to a SPASE resource description and that all URLs resolve to a web page (or service). SPASE has tools to perform both types of checks. On is the on-line validator (http://validator.spase-group.org/) and the other are the command-line tools spase-validate and spase-refcheck.

Single Description

When validating a single SPASE description the on-line validator is a good option. You can upload the file, perform validate and get detailed information on any errors. The on-line validator can also perform referential checks. The command-line tools can also validate and perform a referential check on a single file. The command to perform a validation on the file "description.xml" is:

```
spase-validate description.xml
```

To perform a referential check on the same file the command is:

```
spase-refcheck -i -u description.xml
```

The "-i" option is to perform a referential check on SPASE resource IDs and the "-u" option is to perform checks on URLs.

Large Collections of Descriptions

With large collections of SPASE description, such as would likely be the case for all the data from an instrument on a space mission, using the on-line validator could be tedious since you need to upload each file one-by-one. With large collections it is better to use the command-line tools because they can recursively scan a folder and check every resource description. To recursively san a folder add "-r" option to the command. To validate all the SPASE description in the folder "mission" use the command:

```
spase-validate -r mission
```

To perform a referential check on the same file the command is:

```
spase-refcheck -r -i -u mission
```

Sharing

One of the main goals of the SPASE community is to openly share metadata. The SPASE community currently uses GitHub (https://github.com) to manage and share metadata. On the GitHub platform files are stored in a repository and repositories can be grouped under an organization. The SPASE has created an "hpde" organization for the Heliophysics Data Environment (HPDE) and each naming authority registered with the SPASE Group (https://spase-group.org/services/naming-authority.html) has a corresponding repository. These repositories are public and anyone can download a copy (with git this is called a "clone"). To create a read-only clone use a command like:

```
git clone https://github.com/hpde/{Naming Authority}.git
```

where {Naming Authority} is replaced with the tag for the naming authority. For example, "NASA".

If you are a metadata provider you will need to have a GitHub account and be added as a contributor the repository. A contributor can write (push updates) to the repository. To be able to push to the repository you need to clone the repository with the SSH method and the command will look like:

```
git clone git@github.com:hpde/{Naming Authority}.git
```

where {Naming Authority} is replaced with the tag for the naming authority. For example, "NASA".

When cloning a repository with the example commands the copy is placed in the current directory in a folder with the name of the {Naming Authority}. You can then edit or add files in the folder.

To push any changes back to the GitHub repository first you must add any new files placed in the folder to the staging area of the repository. The command to use (inside the repository folder) is:

```
git add .
```

Then, to add any changed files to the staging area and create a snapshot of all the changes use the command (inside the repository folder):

```
git commit -a -m "Brief message describing changes."
```

Now you are ready to share the changes by placing them on the GitHub repository. This is performed with a push. The command is:

```
git push
```

After the push is complete anyone can get the updates with either a clone, or if you already have a clone, using a pull request to update your local copy. The command to do a pull is:

```
git pull
```

When collaborating with others on a repository it is allows best to perform a "git pull" prior to make changes to local copy. This will make your local copy match the current contents of the GitHub repository.

Landing Pages

One service provided to the community are "landing pages". These are web pages which are a formatted version of the SPASE descriptions in the GitHub repositories. In addition to the web page are XML and JSON versions of the SPASE descriptions. The landing pages can be accessed at "https:// hpde.io". These pages are generated on a regular schedule and can be used as the landing page for a DOI created for the resource.

DOI Minting

Publications typically require a DOI (Digital Object Identifier) when referencing data or documents in paper or article. A DOI along with essential publication information may be included in a SPASE description. The SPASE group has the ability to mint (create) a DOI if you do not already have one for the resource. Whether you mint the DOI or have the SPASE group mint one for you there is a tool in the SPASE Resource Toolkit which can create a DOI generation request. The tool is "spase-doi-request". If the PublicationInfo element of a SPASE description is populate, then you will have a fully formed DOI request. This request is in the XML form that DataCite uses so if you have minting ability you can submit it under your own account. Contact the SPASE group for more details.

Harvesting

Harvesting is the process of extracting information from SPASE descriptions for use in other systems. For SPASE descriptions shared openly though GitHub you can acquire copies of SPASE descriptions with a "git clone" command. To get a copy of the current set of descriptions use the command

```
git clone --single-branch --depth=1 \

https://github.com/hpde/{Naming Authority}.git
```

Where {Naming Authority} is one of the repositories found on the list at:

https://github.com/hpde

This will clone (create a copy) of the default branch of the repository and only include the most recent files. To refresh your copy issue the command:

```
git pull
```

from inside the clone.

One example of harvesting information from the SPASE description is creating a request to generate a DOI. Simple run the command:

```
spase-doi-request {file.xml}
```

If you SPASE description contain a completed <PublicationInfo> then the tool will output a DOI request file which can be submitted to a DataCite compliant minting service. There are options that can be supplied to the "spase-doi-request" tool that can specify any missing information.

Tips and Tricks

Follow Conventions

Following the repository organization and file naming conventions makes finding resources easier and several tools (and some services) are designed to work best with these conventions.

Do Validations and Referential Checks before Push

Repositories are managed and updated in four steps. First you create a clone. Second, make changes, additions or perform an updates on the clone. Third, perform validation and referential checks (using spase-validate and spase-refcheck) on the clone. Forth, commit and push changes to the repository. Following these step allow you to catch errors before the files are shared.

Change <ReleaseDate> After Updates

The <ReleaseDate> indicates the date of the most recent change of a SPASE description. It acts like a version number for the description. When making scripted changes to many descriptions, the "space-restamp" can be used to set the <ReleaseDate>. It has a recursive option so it can change multiple files with one command.

Final Thoughts

There are a lot of tools and a lot of existing SPASE descriptions. There are also a variety of services which harvest SPASE description to create search engines, provide uniform access to data and manage a project's collections. It can seem like a lot to take in when starting to use SPASE, which is because it forms a distributed data environment which was built through an accumulation of a lot of smaller efforts. You can contribute to this data environment by creating and sharing SPASE descriptions for the things you are most familiar with. Even a single SPASE description for one data collection, shared through public repositories, will enhance the data environment and enable better research. Start small and become a part of something bigger.