Oxide Usage

Definitions

OID
: (Oxide ID) refers to a unique object in the system, usually a file.

Collection
: a permanent set of objects that are assigned a name by the user and have an ID.

CID
: (Collection ID) a type of OID that refers to a collection.

Context
: a temporary, non-persistant set of objects that are easy to reference in the shell.

Module
: a function that can be run over an OID or set of OIDs and caches the results in the datastore.

Plugin
: a set of functions called from the shell that interact with the user.

Scratch Directory
: the directory /scratch off of the Oxide root directory, frequently used when outputting files or for temporary files.

Referring to Objects in the System

Use special characters to tell the Oxide shell what you are referring to:
% oid
& collection_name
$ context_item
@ variable_name
ˆfile_name

Using Help

`help` by itself shows the built-in commands and the special reference characters.
`help` followed by a command name gives help for that command.
`help` followed by a loaded plugin name (use `show plugins` to see loaded plugins) gives help for the entire plugin.
`help` followed by a plugin function name provides help for that function.
If this is a serious emergency, call 911.

Using Show

Pipe the output of plugin functions or modules to `show` to display whatever they return. `show` has a few additional built-in features:

```
show collections
show collections --verbose
```

shows a list of all the collections and some information about them.

```
show context
show context --verbose
```

shows the current context with numbering and some meta-data.

```
show plugins
```

shows the currently installed plugins.

```
show modules
```

shows the currently available modules.

Importing Files and Making Collections

When importing files or a directory, always make a collection, so that you have an easy way to refer to the files.
Example:

```
import datasets/sample | collection create sample
```

imports all of the files in the datasets/sample directory and creates a collection named sample.

Working with Contexts

Contexts are temporary sets of files that you want to work with. Collections are permanent objects in the system that are have a unique ID.
To create a context, you can use the `set` subcommand and pass it a collection. For example:

```
context set &sample
```

You can then use `show` to see the numbers assigned to the context and use the $ to refer to individual items in the context. Contexts also support slicing.

Piping

Piping is used to pass data between commands in Oxide. Whatever is emitted by the preceding command is appended to the argument list of the following command. For example, the following pair of commands are equivalent:

```
context set &sample
&sample | context set
```

This is useful for stringing together commands that operate on OIDs by either filtering or creating new objects. More examples are given in the shell tutorial.

Working with Plugins

Load a plugin using `plugin` followed by the plugin name (tab complete will provide a list if it is working on your system). The functions available in the plugin will now be available as commands in the shell. The default plugin is loaded automatically on starting the shell. See the list of available plugin commands provided by the currently loaded plugins using `show plugins`.
Some commonly used plugins:

unpack provides functions to unpack various packed and compressed formats.

re_tools provides functions to reverse engineer Windows PE files.

bin_tools provides functions to examine and compare binary file formats

string_tools provides a set of utility functions for playing around with strings, xoring, packing, etc.

Working with Modules

Modules are typically interacted with indirectly by using plugins. To run a module directly, use the `run` command followed by the module name and some OIDs. Modules do not print to the screen, so pipe the output to `show` if you want to see the result.
Example:

```
run byte_histogram &sample | show
```

See a list of available modules using `show modules`.