Revolver Quick Start Guide

Welcome to the Revolver quick start guide. This document is intended to get you off the ground and running quickly.

HELP

Help is built into Revolver, so after you open a command prompt just type `help`. This will list the available commands and a brief description of each. To get help on any of the commands listed type `help <cmd>` where `<cmd>` is the command name. This gives to the command description along with the command usage, parameters definition, comments and examples of how to use the command.

COMMAND SYNTAX

A command entered on the command line usually consists of the command name, any parameters to pass to the command, and an optional path to operate on instead of the current context path.

```
sf -nv -f text Boo! /Sitecore/content/home
```

In the above command, `sf` is the name of the command to execute and we're passing `-nv`, `-f`, `text` and `Boo!` as parameters and also providing a path. The path can be either absolute or relative to the current context item.

Revolver uses spaces to separate arguments entered on the command line. For this reason you can't drop space when you think they aren't required as they are a delimiter. If you want to use spaces in a parameter, enclose the entire parameter in parenthesis. So if I wanted to pass a parameter "Welcome to Sitecore" I would enclose it as in the following example.

```
sf -f text (Welcome to Sitecore)
```

MOVING AROUND

The command prompt environment is much like the command shell of your OS but instead of a file system to navigate round, we have the content tree. You can also think about the databases like different drives.

To change the content item we're focused on use the `cd` command.

```
cd content
```

Paths provided to `cd` can be relative or absolute. You can also specify a database at the start of the path to change databases and paths at the same time.

```
cd /core/sitecore/content/applications
```

We can just change the database and move to the root item of that database by using the `cdb` command.

```
cdb extranet
```

The `cd` command also accepts a slash (/) to refer to the root node (sitecore node).

To get a list of all of the children of the current item and show the possibilities for navigating around use the `ls` command.

```
ls
ls -r news
ls -r news -c
```

The `-r` parameter allows you to supply a regular expression to match children on. Only those that match will be displayed. By default the regular expression matching is done case-insensitive. The `-c` parameter causes the match to be case sensitive.

By default the `ls` command will list the children in the order specified in the content tree. The `-a` and `-d` parameters allow sorting alphatically and in reverse order respectivly.

```
ls
ls -a
ls -a -d
```

We can also change the current language. Use the `cl` command to do this.

```
cl da
```

To get information about our current context we can use `pwd` to get the current path, `pwdb` to get the current database and `pwl` to get the current language. This context information can also be set to display in the command prompt itself. For more information on the command prompt see the section "The Prompt".

GETTING INFORMATION

We can view the current fields of an item using the `gf` command. This command will list all fields defined for an item. We can specify a particular field by name to retrieve by using the `-f` parameter.

```
gf
gf -f title
```

The `ga` command allows access to the items attributes and works in the same way as the `gf` command, except the parameter to use for a particular attribute is `-a`.

```
ga
ga -a id
```

SETTING INFORMATION

We can also update content using Revolver. To set a field use the `sf` command passing the name of the field and the new value.

```
sf title Hello
```

This will also cause a new version of the item to be created before the update occurs. If you don't want to create a new version, pass the `-nv` parameter to `sf`.

```
sf -nv text (Welcome to Sitecore)
```

If you wish to enter a new line character into a field use shift + enter. This causes Revolver to use the newline as part of the command rather than execute the command.

In the same fashion as setting fields, we can also set attributes with the sa command.

```
sa templateid {76036F5E-CBCE-46D1-AF0A-4143F9B557AA}
```

OPERATING ON MULTIPLE ITEMS

Revolver contains 2 commands to execute an arbitrary command on a set of items. These are find and query.

Find operates on either children or descendents of the context item and uses filters to filter the inclusion list down.

```
find pwd
find (ga -a id)
find -r -t {76036F5E-CBCE-46D1-AF0A-4143F9B557AA} pwd
```

The first example above will print the full path of children of the current item. The second example will list the IDs of the children of the current item. The last example operates on descendents due to the `-r` parameter, of the current item. It then filters by template due to the `-t` and GUID parameters, then executes `pwd` on the resulting item list, which will print the full path. So the command has the affect of printing the full path of all descendents of the current item based on the given template ID.

Query allows selecting an item list based on a Sitecore query.

```
query child::* pwd
query /Sitecore/content/descendant::*[@@templateid='{76036F5E-CBCE-46D1-AF0A-4143F9B557AA}'] (ga -a id)
```

Both the above commands support the `-ns` flag meaning "no statistics". This prevents the command from outputting how many items were found. This flag is used when you want to search for a particular item and feed the output of the command to another command.

```
cd < (query -ns (*[startswith\(@@key,"c"\)]) pwd)
```

SUBCOMMANDS

Subcommands allow the output of one command to be used as a parameter to another command. The subcommand is evaluated and the output placed into the appropriate parameter.

```
find -t < (ga -a templateid) pwd
```

The left angle bracket denotes that the next parameter is a subcommand and should be evaluated and not passed verbatim to the command.

In the above example the template ID of the current item is evaluated and passed to the template filter parameters of the `find` command. This would have the affect of finding all children based on the same template as the current item and printing the full path of each matched item.

Multiple subcommands can be used in a single command, and they can be nested as well, so subcommands can contain subcommands themselves.

```
echo < (ga -a id) : < (ga -a name)
echo < (replace home house < (ga -a name))
```

The `echo` command above prints whatever is passed to it onto the command line. This is handy for output text from a script. The first example above can be combined with a `find` command to print the IDs and names of all children of the current item.

```
find (echo < (ga -a id) : < (ga -a name))
```

MANIPULATING LISTS

Many fields in Sitecore store their data as pipe seperated ID lists. The lm command is used to help manipulate these lists by providing parameters to add or remove specific IDs from the list.

```
lm -a {ID1}|{ID2} | {ID3}
lm -r {ID1}|{ID2} | {ID2}
```

The `-a` parameter indicates adding an element and the `-r` parameter indicates removing an element. You also need to provide the list itself, the list separator and the element to add or remove.

This command is only a string manipulator and you'll have to utilise subcommands to actually change a field value. The following command removes a specific ID from the "Related Articles" field of the context item.

```
sf (related articles) < (lm -r < (gf -f (related articles)) |
{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9})
```

FILES

Echo is also used to direct output to a file, or read content from a file. Using the `-f` parameter, we can direct the output to either a relative or absolute file.

```
echo -f output.txt (this is some output)
echo -f c:\temp\output.txt < gf
```

Relative files are saved to the temp folder. The above commands will write a new file each time. If we wish to append a file, use the `-a` parameter.

```
echo -a -f output.txt < ga
```

The content of a file can also be read using the `-i` parameter.

```
echo -i -f  c:\temp\input.txt
```

The above command would have the affect of outputting the contents of the given file input to the command window. These 2 uses of echo are useful for saving an item to file, then recreating it from file, which comes in handy with testing.

To save an item to file, use `gi` with `echo`.

```
echo -f c:\temp\item.xml < gi
```

To create an item from file, use `touch` with `echo`.

```
touch -x < (echo -i -f c:\temp\item.xml)
```

ITEM CREATION AND RELOCATION

To create an item we use the `touch` command and can pass to it either the template, master (Sitecore 5.3.x) or branch (Sitecore 6+) we wish to base the item on. To create a new item called newItem from the "document" template:

```
touch -t document newItem
```

To delete an item use the `rm` command.

```
rm newItem
```

By default the `rm` command will move the item to the recycle bin. If you want to permanently delete the item instead of recycling it, pass the `-nr` parameter.

```
rm -nr item
```

To copy an item to a new location use the `cp` command, passing the path to copy the item to and optionally a new name for the new item. Keep in mind the ordering of the parameters here which is the reverse of a normal OS shell command for copying a file. Instead of copy from to, Revolver follows the pattern of the other commands for changing the context upon which it's operating, so the copy command is copy to optional from. If "from" is not provided the current context is used.

```
cp subfolder/something/newItem
cp subfolder/something
cp ../../newItem subfolder/oldItem
```

If you want to copy the entire subtree instead of a single item, use the –r parameter as well.

```
cp –r folder/newItemTreeName
```

We can also move items using the `mv` command. Just provide the command with the new path to move the item to.

```
mv subfolder
```

CHANGE ITEMS

The `ct` command can be used to change an item's template.

```
ct (MyProject/News Article)
ct {76036F5E-CBCE-46D1-AF0A-4143F9B557AA} /sitecore/content/item
```

The ct command can accept either a path relative to the templates folder or the ID of the template to change the item to.

By default this command will fail if you try to change the item's template to an incompatible template; one that does not contain all the currently used fields. In such as case you can use the -f parameter to force the change and accept that you will be losing field data. Note that the new template does not need to contain all the fields the original template had, only the ones that have content in them.

ENVIRONMENT VARIABLES

Temporary information can be stored in environment variables. This mechanism can also be used to exchange or set information for commands. To get a list of the current environment variables use the `set` command with no parameters. Doing this, you'll see some default environment variables detailed in the following table.

Variable	Description
prompt	Controls the prompt displayed at the start of a command prompt line
prevpath	Contains the previous context path to the current path
outputbuffer	Lengh (in characters) of text to be displayed in the output window

To use an environment variable, just enclose the variable name in dollar signs.

```
cd $prevpath$
```

Just remember that if your path might contain spaces you'll have to enclose the variable in parenthesis.

```
cd ($prevpath$)
```

To create an environment variable pass the name of the variable and it's value to the `set` command.

```
set myVar Hello
set id < (ga -a id)
```

To overwrite a variable, use the same syntax with `set` as for creating the variable.

Environment variables can be cleared by passing the name of the variable, but no value to `set`.

```
set id
```

THE PROMPT

The `prompt` environment variable is used as a template for the actual command prompt which is output at the start of each input line. The prompt can contain any characters you like but must end with a right angle bracket. The prompt environment variable can also contain some special tokens which get substituted on each evaluation of the prompt. These are listed below.

Token	Description
%path%	Provides the full path of the current item
%itemname%	Provides the name of the current item
%db%	Provides the name of the current database
%lang%	Provides the title of the current language
%date%	Provides the current date
%time%	Provides the current time

Information for the prompt tokens can be viewed within Revolver by executing help on the `prompt` help topic.

```
help prompt
```

The following example sets a prompt containing the current database, item path and language.

```
set prompt (%db%:%path% [%lang%] >)
```

COMMAND CACHE

Revolver stores each command which is executed in the command cache to allow easy retrieval of previously executed commands. To cycle through the command cache just use the up and down arrows.

CREATING AND EXECUTING SCRIPTS

Revolver allows a group of commands to be stored as a script so they can be executed at a later stage without all the rekeying. Scripts are stored in the core database.

To create a script, swap to the core database using the database switcher then open a content editor and navigate to `/sitecore/system/modules/revolver/scripts`. Scripts can be created directly in this location of under any folder structure you choose, but the script should be uniquely named as Revolver will execute the first script found in the event more than 1 script shares the same name.

To execute a script just call it by name at a command prompt.

```
myscript
```

We can also pass parameters into a script. These are available inside the script using the same syntax as environment variables except script parameters are references by index. So if I executed the following line:

```
myscript hello par2
```

Then inside the script I can refer to the 1st parameter (hello) as `$1$` and the 2nd parameter (par2) as `$2$`. So my script might be something like:

```
find -r -f title ($1$) (replace < (gf -f title) ($1$) ($2$))
```

And of course scripts can contain multiple lines as well.

Scripts can also contain comment lines. A comment line must start with hash (#).

```
# this is my script
# parameter 1 is the name to search for
find -r -a name $1$ pwd
```

The script definition item contains fields to allow you to supply help information for scripts so a user using your script can get information on the usage of the script as well as the parameters. It is highly recommended you enter the help fields of the script item.

STARTUP SCRIPTS

There are many reasons why you might want a script to execute when you open a new instance of Revolver; to define your prompt or bind in custom commands or rebind existing commands to a different moniker. After you've created the script you want to run on startup you can create a user script in the core database at `/sitecore/system/modules/revolver/startup`. The user script contains fields for the script to execute, a comma separated list of roles and a comma separated list of users. To execute the script for someone with a particular role, enter the role name in the role field. To execute the script for a user, enter the username in the user field.

REBIND COMMANDS

The names used for the commands in Revolver are just monikers bound to a class in the `Revolver.Core.Commands` namespace. If you would rather use a different command name you can rebind the command using the bind command.

To get a list of the current bindings, execute `bind` with no arguments.

To rebind a command pass the class and the new moniker to `bind`.

```
bind list dir
```

It is handy to use a startup script to set your own bind list.

If you want to reset the bindings to the default monikers, just pass the `-r` parameter.

```
bind -r
```

CUSTOM COMMANDS

Bind can also be used to bind in custom commands to the Revolver shell. Your custom class must implement the `Revolver.Core.Icommand` interface. The `Run` method is called when the command is executed at the command prompt. The `Initialise` method is called just before `Run` and gives the class author the chance to store a reference to the context object and the format context. The context object contains the current context information like the current database, item and language. The format context contains information and utility methods for the current UI like the new line character sequence.

The following is an example of a custom command that has been compiled into an assembly called `CustCommand`.

```
using System;
using System.Collections.Generic;
using System.Text;
using SCCli.Core;

namespace CustCommand {
      public class Time : Icommand
            public void Initialise(Context context, IformatContext
formatContext) {}

            public CommandResult Run(string[] args) {
                  return new CommandResult(CommandStatus.Success,
DateTime.Now.ToString());
            }

            public string Description() {
                  return "Provide the current date and time";
            }

            public HelpDetails Help() {
                  HelpDetails details = new HelpDetails();
                  details.Description = Description();
                  details.Usage = "<cmd>";
                  return details;
            }
      }
}
```

This custom command can now be bound into the shell using the following:

```
bind CustCommand.Time,CustCommand time
```

Now I can invoke the custom command by calling the moniker:

```
time
```

SEARCHING IN OUTPUT

Sometimes the execution of a command may result in a large number of output lines. Revolver provides the `grep` command to allow searching for lines in the output that match a regular expression. This command works similarly to the GNU command of the same name, except you can't pipe output into a command. Instead you'll use a subcommand.

```
grep myitem < ls
grep publish < ps
```

By default the regular expressions comparison is case insensitive. Use the `-c` parameter to use a case sensitive match instead.

OTHER COMMANDS

We have so far covered the most commonly used commands although Revolver contains many more. The following table lists the remaining commands and a brief description of what each does. Remember, you can always view the full list of commands available in Revolver by typing `help`.

Command	Description
gi	Get the XML which constitutes an item or item tree.
ci	Check in item (remove locks I hold)
co	Check out item (obtain a lock on the item)
cpp	Copy presentation to a new item or between devices
rmv	Purge previous versions of the item
replace	Perform string replacement manipulation using regular expressions
split	Split a string and iterate over the results
ps	List the current jobs of a system
cpl	Copy fields of one language to another language within the same item
val	Perform validation defined on the items template
timer	Display how long a command takes to run
users	Show current user sessions and allow terminating of sessions
lsdb	List the databases
links	Display references to and from this item from the link database

EXIT REVOLVER

To exit Revolver either close the window or type "exit" at the prompt.