Guide to API Privilege Escalation - Call Your Application

Introduction

To use your `AdminBin` application, call the appropriate method.

The method to use depends on the target cPanel & WHM version, and whether you used the `Cpanel::AdminBin::Script::Call` module.

Application call methods

Select a tab below to view the recommended method for your target cPanel & WHM versions.

> **Note:**
>
> cPanel, L.L.C. introduced this functionality in cPanel & WHM version 54 for admin modules in Perl.

To use your `AdminBin` application, call the `Cpanel::AdminBin::Call::call` function. This function passes your arguments to the `AdminBin` module. It is intended only for those admin modules that subclass the `Cpanel::AdminBin::Script::Call` class; to call other admin modules, use The Standard Method.

Example

The following example demonstrates how to use this functionality within a Perl script:

Related documentation

- Guide to Standardized Hooks - Privilege Escalation — Scripted hook action code that runs as the cPanel user can escalate that user's privileges.
- Guide to API Privilege Escalation — To run a function with escalated privileges, call a function through the `Call` method or use the `send_cpwrapd_request` pluggable wrapper.
- Guide to API Privilege Escalation - Legacy Method — In cPanel & WHM version 11.36 and earlier, you **must** write the scripts to allow functions to run with elevated privileges.
- Guide to API Privilege Escalation - Configuration Files — The configuration file defines two configuration settings that determine your application's behavior.
- Guide to API Privilege Escalation - Application Files — The application file contains the functions that you wish for the system to escalate.

```
#!/usr/local/cpanel/3rdparty/bin/perl

use Data::Dumper ();

use Cpanel::AdminBin::Call ();

sub do_MyExample_stuff {
    my $thing_to_do = shift;
    my $string_to_mess_with = shift;

        #Prevent potential action-at-a-distance bugs.
        #(cf. documentation for CPAN's Try::Tiny module)
        local $@;

        my $val;

        #NOTE: call() will throw an exception if there was an
error
        #while communicating with the admin module or if an
exception
        #escaped from a method call within the admin module.
        my $ok = eval {
                $val = Cpanel::AdminBin::Call::call(
                        'MyNamespace',
                        'MyExample',
                        $thing_to_do,
                        $string_to_mess_with,
                );

                1;
        };

        if ($ok) {
                return ref($val) ? Data::Dumper::Dumper($val)
: $val;
        }

        my $err = $@;
        return "Error: $err";
}

print "Content-type: text/html\r\n\r\n";

print "<pre>";

print "ECHO test:\n" . do_MyExample_stuff("ECHO","Hello,
World!") . "\n\n";
print "MIRROR test:\n" . do_MyExample_stuff("MIRROR","Hello,
World!") . "\n\n";
print "BOUNCY test:\n" . do_MyExample_stuff("BOUNCY","Hello,
World!") . "\n\n";
print "HASHIFY test:\n" . do_MyExample_stuff("HASHIFY","
Hello, World!") . "\n\n";
print "WRONG test:\n" . do_MyExample_stuff("WRONG","Hello,
World!") . "\n\n";

print "DEATH test:\n" . do_MyExample_stuff("DEATH") . "\n\n";

print "test complete!\n";
```

In the same directory as the `MyExample` module file, you **must** create the `MyExample.conf` configuration file with the following contents:

```
mode=full
```

For more information about the configuration file, read our Configuration Files documentation.