

# Programmatic Control of Dynamic Plugins

---

The loading of external dynamic filters can be controlled during runtime with an environment variable, `HDF5_PLUGIN_PRELOAD`, but this variable offers no option of control from within a program built on the library. The recommendation to change the state of the global plugin variable from negative logic, disable plugins, to positive logic, enable plugins is discussed in this document.

---

**Introduced with**

**HDF5-1.8.15**

**in**

**May 2015**



---

## Copyright Notice and License Terms for HDF5 (Hierarchical Data Format 5) Software Library and Utilities

HDF5 (Hierarchical Data Format 5) Software Library and Utilities  
Copyright 2006-2015 by The HDF Group.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities  
Copyright 1998-2006 by the Board of Trustees of the University of Illinois.

### All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution.
3. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change.
4. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by The HDF Group and by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and credit the contributors.
5. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively.

**DISCLAIMER:** THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no event shall The HDF Group or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois, Fortner Software, Unidata Program Center (netCDF), The Independent JPEG Group (JPEG), Jean-loup Gailly and Mark Adler (gzip), and Digital Equipment Corporation (DEC).

Portions of HDF5 were developed with support from the Lawrence Berkeley National Laboratory (LBNL) and the United States Department of Energy under Prime Contract No. DE-AC02-05CH11231.

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.

**DISCLAIMER:** This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

## Contents

1. Introduction .....	4
2. Use Cases .....	5
3. Implementation .....	6
4. Revision History .....	9

## 1. Introduction

The loading of external dynamic filters can be controlled during runtime with an environment variable, `HDF5_PLUGIN_PRELOAD`. However, it offered no option of control from within a program built on the library.

The environment variable can control the loading of dynamic filters at runtime, but it will disable it for all running programs that access that variable using the library. It is expected that the environment variable will be set before any programs execute and remain constant throughout the programs' execution life.

The need for finer grained control of the feature was exposed when HDF5 tools were built with the `static-exec` option and attempted to use dynamic filter loading. Because the tool and the filter used different runtime instances an exception was raised when a different C runtime tried to free the memory allocated by the other C runtime instance.

Another recommendation was to change the state of the global plugin variable from negative logic, disable plugins, to positive logic, enable plugins.

## 2. Use Cases

1. Disable all plugins - `H5PLset_loading_state (0)`
2. Enable all plugins - `H5PLset_loading_state (-1)`
3. Disable plugin X - requires user to negate the state with a 0 in bit position X and AND it with the result from a `H5PLget_loading_state` call.

```
H5PLget_loading_state(&curr_setting)
new_setting = curr_setting & ~H5PL_FILTER_PLUGIN
H5PLset_loading_state (new_setting)
```

4. Enable plugin X - requires user to set the state with a 1 in bit position X and OR it with the result from a `H5PLget_loading_state` call.

```
H5PLget_loading_state(&curr_setting)
new_setting = curr_setting | H5PL_FILTER_PLUGIN
H5PLset_loading_state (new_setting)
```

### 3. Implementation

In the `H5PL.c` file in the source folder, the local variable is declared as a signed `int`:

**Source File: H5PL.c**

```
static int      H5PL_plugin_g = -1;
```

This variable is used for both the global disabling all plugins as well as for the individual plugins. If all plugins should be disabled, the `H5PL_plugin_g` should be zero. If this variable is non-negative then the value of the individual bits control the individual plugins. The plugin bit will correspond with the `H5PL_type_t` enum value for that plugin. For filters, bit 0 will enable filter plugins if set to 1.

In the `H5PLextern.h` file is the existing declaration of the `H5PL_type_t` enum:

**Source File: H5PLextern.h**

```

/*****
/* Public Typedefs */
/*****
/* Plugin type */
typedef enum H5PL_type_t {
    H5PL_TYPE_ERROR      = -1, /*error          */
    H5PL_TYPE_FILTER     = 0, /*filter         */
    H5PL_TYPE_NONE       = 1  /*this must be last! */
} H5PL_type_t;

```

The functions use one argument to enable or disable the individual plugins. The function need to check the environment variable method of disabling the use of dynamic filter loading in order to not override the environment setting. The argument directly sets/unsets the global `H5PL_plugin_g` variable.

**Source File: H5PL.c**

```

/*-----
* Function:   H5PLset_loading_state
*
* Purpose:   Control the loading of dynamic plugins.
*
*           This function will not allow plugins if the pathname from
*           the HDF5_PLUGIN_PRELOAD environment variable is set
*           to the special ":@" string.
*
*           plugin bit = 0, will prevent the use of that dynamic plugin.
*           plugin bit = 1, will allow the use of that dynamic plugin.
*
*           H5PL_TYPE_FILTER changes just dynamic filters
*           A negative value will enable all dynamic plugins
*           A zero value will disable all dynamic plugins
*
* Return:   Non-negative or success

```

```

*
*-----
*/
herr_t
H5PLset_loading_state(int plugin_flags)
{
    char *preload_path;
    herr_t ret_value = SUCCEED; /* Return value */

    FUNC_ENTER_API(FAIL)

    /* check for global setting first */
    if(plugin_flags < 0)
        plugin_flags = -1;
    /* change the bit value of the requested plugin(s) */
    H5PL_plugin_g = plugin_flags;
    /* check if special ENV variable is set and disable all plugins */
    if(NULL != (preload_path = HDgetenv("HDF5_PLUGIN_PRELOAD"))) {
        /* Special symbol "::" means no plugin during data reading. */
        if(!HDstrcmp(preload_path, H5PL_NO_PLUGIN))
            H5PL_plugin_g = 0;
    }
}
done:
    FUNC_LEAVE_API(ret_value)
} /* end H5PLset_loading_state() */

/*-----
* Function:    H5PLget_loading_state
*
* Purpose:    Query state of the loading of dynamic plugins.
*
*             This function will return the state of the global flag.
*
* Return:    Zero if all plugins are disabled, negative if all
*             plugins are enabled, positive if one or more of the plugins
*             are enabled.
*-----
*/
herr_t
H5PLget_loading_state(int* plugin_flags)
{
    herr_t ret_value = SUCCEED; /* Return value */
    FUNC_ENTER_API(FAIL)

    *plugin_flags = H5PL_plugin_g;
done:
    FUNC_LEAVE_API(ret_value)
} /* end H5PLget_loading_state() */

```

To support the flags parameter, the following convenience defines help because the bit position defines in H5PL\_type\_t enum start at 0.

**Define Value**

```

/* Common dynamic plugin flags */
#define H5PL_FILTER_PLUGIN    0x0001

```

Also the `H5PL_no_plugin` function has been removed as it was never used. The only use of `H5PL_plugin_g` variable is in the `H5PL_load` function.

### Private `H5PL_load` function in `H5PL.c`

```
const void *
H5PL_load(H5PL_type_t type, int id)
{
    htri_t    found;                /* Whether the plugin was found */
    const void *plugin_info = NULL;
    const void *ret_value = NULL;
    FUNC_ENTER_NOAPI(NULL)
    /* Check for "plugins are disabled" indication */
    if(H5PLplugin_g == 0)
        HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required dynamically
loaded plugin '%d' is not available globally", id)
    switch (type) {
    case H5PL_TYPE_FILTER:
        if((H5PL_plugin_g & H5PL_FILTER_PLUGIN) == 0)
            HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required
dynamically loaded filter plugin '%d' is not available", id)
        break;
    default:
        HGOTO_ERROR(H5E_PLUGIN, H5E_CANTLOAD, NULL, "required dynamically
loaded plugin '%d' is not available", id)
    }
    /* Initialize the location paths for dynamic libraries, if they aren't
    * already set up.
    */
    ...
}
```

### Public Header File

```
H5_DLL herr_t H5PLset_loading_state(int plugin_flags);
H5_DLL herr_t H5PLget_loading_state(int* plugin_flags/*out*/);
```



## 4. Revision History

*May 2015*

Version 1.