

Getting Settable Parameter Values in NetWare

Novell Cool Solutions AppNote

www.novell.com/coolSolutions

JULY 2004

Russell Bateman
Senior Software Engineer
Server-library Development
rbateman@novell.com

This How-to AppNote discusses how to obtain the values of command-line-settable parameters from a NetWare-loadable Module (NLM) and explores creating and maintaining your own settable parameters.

Contents

- Introduction and Interfaces
- Getting Settable Parameter Values from NetWare
- Managing Your Own Settable Parameters for NetWare
- Conclusion

Topics	NLM development, kernel and low-level code
Products	NetWare 5, NetWare 6, NetWare 6.5
Audience	Developers
Level	Advanced

INTRODUCTION AND INTERFACES

Many aspects of NetWare are configured using a “set command” facility that has been present on NetWare since the beginning. The current state or value of set parameters on NetWare can be very useful to an NLM application.

The header, *netware.h*, prototypes the interfaces for this feature. The original function signatures were misspelled and have remained that way. The header uses the C preprocessor to correct this. I say this only because if you go into the system debugger and type

```
# b=GetSettableParameterValue
```

you will be disappointed because the entry point is really *GetSettableParameterValue*.



It is possible to set as well as get a set parameter, but you must be very careful in setting it. Experiment and test to make certain you do not crash the server. Here are all the prototypes:

```
int GetSettableParameterValue( int slot, const char *name, void
    *value );
int ScanSettableParameters( int scanCategory, uint32_t
    *scanSequence, const char *name, int *type, uint32_t *flags,
    int *category, void *description, void *value, int
    *lowerLimit, int *upperLimit );
int SetSettableParameterValue(int slot, const char *name, void
    *newValue);
```

I have never used the scan function and I don't know who, if anyone, has, so I'm not going to talk about it. Look at the documentation at <http://developer.novell.com> instead.

The *slot* argument actually holds the connection to NetWare the NLM has. I use 0 consistently. Since LibC doesn't accept or give out connection slots as CLib does and as 0 always works, it seems pointless to go beyond this explanation except to say that I don't believe that using a non-zero connection slot belonging to an actual, authenticated object has any effect upon the success of failure of these calls.

GETTING SETTABLE PARAMETER VALUES FROM NETWARE

For getting a value, you have to know what data type a value is. If it is a string, then you pass a pointer to a buffer (*value*) with ample room to handle anything that will get copied into it. If an integer, then to a 4-byte wide integer.

String Data

For example, in the library time code, I have to find out what the rules for time zones in effect are:

```
char    buffer[80];
err = GetSettableParameterValue(0,
    "Start of Daylight Savings Time", buffer);
```

The set parameter is represented by an ASCII string containing the exact wording of the parameter. It is case-insensitive.

If you lived in my locale, the following response would come back:

```
"(April Sunday First 2:00:00 AM)"
```

I parse this into a rule formalism in order to calculate whether a given date and timestamp, say one passed to ANSI function `localtime`, is on summer time or not.

Integer Data

Elsewhere, in CLib for example, I get the daylight offset in this manner:

```
int    daylightOffset;
err = GetSettableParameterValue(0,
    "Daylight Savings Time Offset", &daylightOffset);
```

This offset is recorded in seconds off standard time, usually 3600 (1 hour).

MANAGING YOUR OWN SETTABLE PARAMETERS FOR NETWARE

It is just as useful to create, initialize and dispose of new settable parameters for NetWare as it is to read the ones it already has. Many of the existing ones are created by loaded NLMs. This is how the monolithic CLib in the days of NetWare 4.11 handled whether to track memory allocations or not.

First, we start by allocating a resource tag.

```
#include <netware.h>

#define NAME          "CLIB Memory Checking"
#define DESCRIPTION  "NLM Library memory allocation system integrity checking"\
                    " and allocation tracking."

int                  gMemTrack = FALSE;
rtag_t              gMemTrackRTag = (rtag_t) NULL;
settableparms_t     gMemTrackStruct;

void SetUpCLibDebugParameters( void )
{
    gMemTrackRTag = AllocateResourceTag(getnlmhandle(),
                                       "CLIB Memory Tracking",
                                       SettableParameterSignature);

    memset(&gMemTrackStruct, 0, sizeof(gMemTrackStruct));

    gMemTrackStruct.value          = &gMemTrack;
    gMemTrackStruct.rTag          = gMemTrackRTag;
    gMemTrackStruct.name          = NAME;
    gMemTrackStruct.type          = SP_TYPE_BOOLEAN;
    gMemTrackStruct.flags        = SP_HIDE;
    gMemTrackStruct.category      = SP_MISCELLANEOUS;
    gMemTrackStruct.callback      = NULL;
    gMemTrackStruct.description   = DESCRIPTION;

    err = RegisterSettableParameter(&gMemTrackStruct);
}
```

Once this is done, the new set command can be used to turn memory tracking on and off:

```
MYSERVER: Set CLIB Memory Checking ON
```

The *value* field of the structure points to the place in memory where the state of the registered settable parameter will be recorded and examined. Remember that this place must be absolutely reliable as commands issued from the console will access it whether or not your NLM ever does. Don't drop your NLM without removing your settable parameter formally.

Note that that ON and OFF, TRUE and FALSE, etc. are interpreted as 1 and 0 and set into the space provided, here, *gMemTrack*. You should never use any other category than SP_MISCELLANEOUS unless you know exactly what you are doing. I believe that a type of SP_TYPE_STRING makes the *value* field of the structure *char**, while SP_TYPE_NUMBER makes it *int* or *long*, etc. At least, this seems obvious even though I cannot lay claim ever to have used it.

Neither I have never used the *lower_limit* or *upper_limit* fields of the *settableparms_t* structure. I believe they are somehow related to the scan function.

Call *DeRegisterSettableParameters* when your NLM unloads and the set parameter will no longer be there. If you do not do this, the OS will think the location pointed at by *value* in the register structure is valid and will dereference it resulting in getting unstable and unrelated values, overwriting data that this address wrongly references, or even page faulting.

```
void RemoveCLibDebugParameters( void )
{
    DeRegisterSettableParameters(&gMemTrackStruct);
}
```

CONCLUSION

Settable parameters in NetWare are a useful feature to which few know how to code and fewer still have coded. This article is not written in exhaustive detail, but it should give you enough information to create some useful parameters for debugging or even for active functionality if your application needs it.

Finally, we did not discuss *SetSettableParameterValue* in this article but its use should follow easily from all that is given.