Subject: [PATCH RFC 9/31] net: Implement the per network namespace sysctl infrastructure
Posted by ebiederm on Thu, 25 Jan 2007 19:00:11 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

The user interface is: register_net_sysctl_table and
unregister_net_sysctl_table. Very much like the current
interface except there is an network namespace parameter.

This this any sysctl in the net_root_table and it's
subdirectories are registered with register_net_sysctl
shows up only to tasks in the same network namespace.

All other sysctls continue to be globally visible.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
include/linux/sysctl.h | 7 ++++
include/net/sock.h | 1 +
kernel/sysctl.c | 71 ++-
net/core/sysctl_net_core.c | 5 +++
net/sysctl_net.c | 20 ++++++++++++
5 files changed, 102 insertions(+), 2 deletions(-)

```
diff --git a/include/linux/sysctl.h b/include/linux/sysctl.h
index 8eba2d2..286e723 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -1044,6 +1044,13 @@ struct ctl_table_header * register_sysctl_table(ctl_table * table);

 void unregister_sysctl_table(struct ctl_table_header * table);

+#ifdef CONFIG_NET
+#include <linux/net_namespace_type.h>
+extern struct ctl_table_header *register_net_sysctl_table(net_t net, struct ctl_table *table);
+extern void unregister_net_sysctl_table(struct ctl_table_header *header);
+DECLARE_PER_NET(struct ctl_table, net_root_table[]);
+#endif
+
 #else /* __KERNEL__ */

 #endif /* __KERNEL__ */
diff --git a/include/net/sock.h b/include/net/sock.h
index 5bf6bb5..01a2781 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
```

```
@@ -1414,6 +1414,7 @@ extern void sk_init(void);

 #ifdef CONFIG_SYSCTL
 extern struct ctl_table core_table[];
+DECLARE_PER_NET(struct ctl_table, multi_core_table[]);
 #endif

 extern int sysctl_optmem_max;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 7da313e..ae6a424 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -45,6 +45,7 @@
 #include <linux/syscalls.h>
 #include <linux/nfs_fs.h>
 #include <linux/acpi.h>
+#include <net/net_namespace.h>

 #include <asm/uaccess.h>
 #include <asm/processor.h>
@@ -135,6 +136,10 @@ static int proc_do_cad_pid(ctl_table *table, int write, struct file *filp,
     void __user *buffer, size_t *lenp, loff_t *ppos);
 #endif

+#ifdef CONFIG_NET
+static DEFINE_PER_NET(struct ctl_table_header, net_table_header);
+#endif
+
 static ctl_table root_table[];
 static struct ctl_table_header root_table_header =
  { root_table, LIST_HEAD_INIT(root_table_header.ctl_entry) };
@@ -1059,6 +1064,7 @@ struct ctl_table_header *sysctl_head_next(struct ctl_table_header
*prev)
 {
  struct ctl_table_header *head;
  struct list_head *tmp;
+ net_t net = current->nsproxy->net_ns;
  spin_lock(&sysctl_lock);
  if (prev) {
   tmp = &prev->ctl_entry;
@@ -1076,6 +1082,10 @@ struct ctl_table_header *sysctl_head_next(struct ctl_table_header
*prev)
  next:
   tmp = tmp->next;
   if (tmp == &root_table_header.ctl_entry)
+#ifdef CONFIG_NET
+   tmp = &per_net(net_table_header, net).ctl_entry;
+  else if (tmp == &per_net(net_table_header, net).ctl_entry)
```

```
+#endif
    break;
  }
  spin_unlock(&sysctl_lock);
@@ -1290,7 +1300,8 @@ int do_sysctl_strategy (ctl_table *table,
  * This routine returns %NULL on a failure to register, and a pointer
  * to the table header on success.
  */
-struct ctl_table_header *register_sysctl_table(ctl_table * table)
+static struct ctl_table_header *__register_sysctl_table(
+ struct ctl_table_header *root, ctl_table * table)
 {
  struct ctl_table_header *tmp;
  tmp = kmalloc(sizeof(struct ctl_table_header), GFP_KERNEL);
@@ -1301,11 +1312,16 @@ struct ctl_table_header *register_sysctl_table(ctl_table * table)
  tmp->used = 0;
  tmp->unregistering = NULL;
  spin_lock(&sysctl_lock);
- list_add_tail(&tmp->ctl_entry, &root_table_header.ctl_entry);
+ list_add_tail(&tmp->ctl_entry, &root->ctl_entry);
  spin_unlock(&sysctl_lock);
  return tmp;
 }

+struct ctl_table_header *register_sysctl_table(ctl_table *table)
+{
+ return __register_sysctl_table(&root_table_header, table);
+}
+
 /**
  * unregister_sysctl_table - unregister a sysctl table hierarchy
  * @header: the header returned from register_sysctl_table
@@ -1322,6 +1338,57 @@ void unregister_sysctl_table(struct ctl_table_header * header)
  kfree(header);
 }

+#ifdef CONFIG_NET
+
+static void *fixup_per_net_addr(net_t net, void *addr)
+{
+ char *ptr = addr;
+ if ((ptr >= __per_net_start) && (ptr < __per_net_end))
+  ptr += __per_net_offset(net);
+ return ptr;
+}
+
+static void sysctl_net_table_fixup(net_t net, struct ctl_table *table)
+{
```

```
+ for (; table->ctl_name || table->procname; table++) {
+  table->child  = fixup_per_net_addr(net, table->child);
+  table->data   = fixup_per_net_addr(net, table->data);
+  table->extra1 = fixup_per_net_addr(net, table->extra1);
+  table->extra2 = fixup_per_net_addr(net, table->extra2);
+
+  /* Whee recursive functions on the kernel stack */
+  if (table->child)
+   sysctl_net_table_fixup(net, table->child);
+ }
+}
+
+static void sysctl_net_init(net_t net)
+{
+ struct ctl_table *table = per_net(net_root_table, net);
+
+ sysctl_net_table_fixup(net, table);
+ per_net(net_table_header, net).ctl_table = table;
+
+ INIT_LIST_HEAD(&per_net(net_table_header, net).ctl_entry);
+}
+
+struct ctl_table_header *register_net_sysctl_table(net_t net, ctl_table *table)
+{
+ if (!per_net(net_table_header, net).ctl_table)
+  sysctl_net_init(net);
+ sysctl_net_table_fixup(net, table);
+ return __register_sysctl_table(&per_net(net_table_header, net), table);
+}
+EXPORT_SYMBOL_GPL(register_net_sysctl_table);
+
+void unregister_net_sysctl_table(struct ctl_table_header *header)
+{
+ return unregister_sysctl_table(header);
+}
+EXPORT_SYMBOL_GPL(unregister_net_sysctl_table);
+#endif
+
+
 #else /* !CONFIG_SYSCTL */
 struct ctl_table_header * register_sysctl_table(ctl_table * table,
       int insert_at_head)
diff --git a/net/core/sysctl_net_core.c b/net/core/sysctl_net_core.c
index 176ad08..76f7a29 100644
--- a/net/core/sysctl_net_core.c
+++ b/net/core/sysctl_net_core.c
@@ -125,3 +125,8 @@ ctl_table core_table[] = {
  },
```

```
  { .ctl_name = 0 }
 };
+
+DEFINE_PER_NET(struct ctl_table, multi_core_table[]) = {
+ /* Stub for holding per network namespace sysctls */
+ {}
+};
diff --git a/net/sysctl_net.c b/net/sysctl_net.c
index cd4eafb..359c163 100644
--- a/net/sysctl_net.c
+++ b/net/sysctl_net.c
@@ -54,3 +54,23 @@ struct ctl_table net_table[] = {
 #endif
  { 0 },
 };
+
+DEFINE_PER_NET(struct ctl_table, multi_net_table[]) = {
+ {
+  .ctl_name = NET_CORE,
+  .procname = "core",
+  .mode  = 0555,
+  .child  = __per_net_base(multi_core_table),
+ },
+ {},
+};
+
+DEFINE_PER_NET(struct ctl_table, net_root_table[]) = {
+ {
+  .ctl_name = CTL_NET,
+  .procname = "net",
+  .mode  = 0555,
+  .child  = __per_net_base(multi_net_table),
+ },
+ {},
+};
--
1.4.4.1.g278f
```