
Subject: [PATCH 3/7] SUNRPC: send notification events on pipefs sb creation and destruction

Posted by [Stanislav Kinsbursky](#) on Fri, 28 Oct 2011 14:27:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

They will be used to notify subscribers about pipefs superblock creation and destruction.

Subscribers will have to create their dentries on passed superblock on mount event and destroy otherwise.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
include/linux/sunrpc/rpc_pipe_fs.h | 8 ++++++++
net/sunrpc/rpc_pipe.c               | 32 +++++++++++++++++++++++++++++++++++++
2 files changed, 40 insertions(+), 0 deletions(-)
```

```
diff --git a/include/linux/sunrpc/rpc_pipe_fs.h b/include/linux/sunrpc/rpc_pipe_fs.h
index cf14db9..c1cdb2f 100644
```

```
--- a/include/linux/sunrpc/rpc_pipe_fs.h
+++ b/include/linux/sunrpc/rpc_pipe_fs.h
@@ -44,6 +44,14 @@ RPC_I(struct inode *inode)
     return container_of(inode, struct rpc_inode, vfs_inode);
 }
```

```
+extern int rpc_pipefs_notifier_register(struct notifier_block *);
+extern void rpc_pipefs_notifier_unregister(struct notifier_block *);
+
+enum {
+  RPC_PIPEFS_MOUNT,
+  RPC_PIPEFS_UMOUNT,
+};
+
+extern int rpc_queue_upcall(struct inode *, struct rpc_pipe_msg *);
```

```
struct rpc_clnt;
diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index a717564..8abeb9d 100644
```

```
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -28,8 +28,10 @@
#include <linux/sunrpc/rpc_pipe_fs.h>
#include <linux/sunrpc/cache.h>
#include <linux/nsproxy.h>
+#include <linux/notifier.h>
```

```
#include "netns.h"
+#include "sunrpc.h"
```

```

static struct vfsmount *rpc_mnt __read_mostly;
static int rpc_mount_count;
@@ -41,6 +43,20 @@ static struct kmem_cache *rpc_inode_cachep __read_mostly;

#define RPC_UPCALL_TIMEOUT (30*HZ)

+static BLOCKING_NOTIFIER_HEAD(rpc_pipefs_notifier_list);
+
+int rpc_pipefs_notifier_register(struct notifier_block *nb)
+{
+ return blocking_notifier_chain_cond_register(&rpc_pipefs_notifier_list, nb);
+}
+EXPORT_SYMBOL_GPL(rpc_pipefs_notifier_register);
+
+void rpc_pipefs_notifier_unregister(struct notifier_block *nb)
+{
+ blocking_notifier_chain_unregister(&rpc_pipefs_notifier_list, nb);
+}
+EXPORT_SYMBOL_GPL(rpc_pipefs_notifier_unregister);
+
static void rpc_purge_list(struct rpc_inode *rpci, struct list_head *head,
void (*destroy_msg)(struct rpc_pipe_msg *), int err)
{
@@ -1010,6 +1026,7 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
struct inode *inode;
struct dentry *root;
struct net *net = data;
+ int err;

sb->s_blocksize = PAGE_CACHE_SIZE;
sb->s_blocksize_bits = PAGE_CACHE_SHIFT;
@@ -1027,8 +1044,20 @@ rpc_fill_super(struct super_block *sb, void *data, int silent)
}
if (rpc_populate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF, NULL))
return -ENOMEM;
+ err = blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
+ RPC_PIPEFS_MOUNT,
+ sb);
+ if (err)
+ goto err_depopulate;
sb->s_fs_info = get_net(net);
return 0;
+
+err_depopulate:
+ blocking_notifier_call_chain(&rpc_pipefs_notifier_list,
+ RPC_PIPEFS_UMOUNT,
+ sb);

```

```
+ __rpc_depopulate(root, files, RPCAUTH_lockd, RPCAUTH_RootEOF);  
+ return err;  
}
```

```
static struct dentry *
```

```
@@ -1043,6 +1072,9 @@ void rpc_kill_sb(struct super_block *sb)  
    struct net *net = sb->s_fs_info;
```

```
    put_net(net);  
+ blocking_notifier_call_chain(&rpc_pipefs_notifier_list,  
+    RPC_PIPEFS_UMOUNT,  
+    sb);  
    kill_litter_super(sb);  
}
```
