ESPRITSYSTEMS

Benutzer/Gruppen- und Rechte-System im Esprit-Server

Verwalten von Benutzern und Gruppen.

Vergabe von Standard-Berechtigungen an Benutzer und Gruppen.

Vergabe von Benutzer/Gruppen-Berechtigungen für Filesystem-Pfade.

Rainer Büsch
2017-11-15

Esprit-Version 12.2e

Inhaltsverzeichnis

1 Einführung

Wie die meisten Server besitzt auch der Esprit-Server ein Authentifizierungssystem für Benutzer. Dies umfasst auch die Zuweisung von Benutzern zu Gruppen sowie ein Konzept für die Vergabe von Berechtigungen.

Darüber hinaus kann der Esprit-Server einen Teil seines Filesystems für die Clients zugreifbar machen. Dies geschieht allerdings unter der Kontrolle eines speziellen Pfad-Berechtigungssystems für Filesystem-Pfade.

Dieses Dokument beschreibt das Benutzer- und Rechte-Konzept des Esprit-Servers, sowie die Funktionsweise des Pfad-Berechtigungssystems in seinen wesentlichen Aspekten.

2 Das Benutzer-System

2.1 Schnittstellen des Benutzer-Systems

Das Benutzer-System des Esprit-Servers ist vollständig durch Schnittstellen (Interfaces) definiert und daher grundsätzlich austauschbar. Selbstverständlich existieren von allen Interfaces Default-Implementierungen, mit denen der Server standardmäßig betrieben wird. Sämtliche persistenten Daten werden in entsprechenden Konfigurationsdateien des Servers gespeichert.[1] Die Schnittstellen sind:

- ➔ **User**
  definiert einen Benutzer im Esprit-Server.
- ➔ **Group**
  definiert eine Benutzer-Gruppe im Esprit-Server
- ➔ **UserManager**
  verwaltet die Benutzer (User-Objekte).
- ➔ **GroupManager**
  verwaltet die Benutzer-Gruppen (Group-Objekte).
- ➔ **GroupMemberManager**
  verwaltet die Zuordnung von Benutzern zu Gruppen.

2.1.1 Benutzer (User)

Die *User* Schnittstelle (Siehe Javadoc Dokumentation) definiert die Eigenschaften eines Benutzers im Esprit-System. Die Wichtigsten sind (* = not-null Feld):

- *userName* *
  Dies ist der eindeutige Login-Name des Benutzers. Er kann nach Anlegen des Benutzers nicht mehr verändert werden.
- *uid* *
  Diese Zahl ist neben *userName* ein zweiter Primärschlüssel für das User-Objekt. Sie ist vorgesehen für die Verwaltung von User-Objekten in einer Datenbank.
- *language* *
  definiert die Sprache eines Benutzers. Abhängig von dieser Einstellung präsentiert sich die Client-Benutzeroberfläche übersetzt in die betreffende Sprache.

1 Grundsätzlich ist es möglich, eine völlig andere Implementierung für das gesamte Benutzersystem zu hinterlegen, die z.B. alle persistenten Daten in einer Datenbank pflegt.

- *foreName *, lastName *, email, comment*
  Persönliche Daten des Benutzers
- *isAdmin*
  ist dieses Flag gesetzt, dann werden Berechtigungen für diesen Benutzer nicht mehr geprüft! Er hat also immer alle Rechte.
- *isLocked*
  ist dieses Flag gesetzt, dann ist der Benutzer gesperrt und kann sich nicht mehr einloggen.
- *password*
  Dieses Feld speichert das aktuelle Benutzer-Passwort in verschlüsselter Form. Im Esprit-Server kann konfiguriert werden, ob für einen Benutzer das Passwort Pflicht ist oder nicht. Das allererste Passwort vergibt der Administrator beim Anlegen des Benutzers.
- *createdTs *, changedTs*
  Zeitstempel für Anlegen und letzte Änderung des User-Objekts.

2.1.2 Gruppe (Group)

Die *Group* Schnittstelle (Siehe Javadoc Dokumentation) definiert folgende Eigenschaften einer Gruppe im Esprit-System (* = not-null Feld):

- *groupName **
  Dies ist ein eindeutiger Name für die Gruppe. Er kann nach Anlegen der Gruppe nicht mehr verändert werden.
- *gid **
  Diese Zahl ist neben *groupName* ein optionaler zweiter Primärschlüssel für das Group-Objekt. Es ist vorgesehen für die Verwaltung von Group-Objekten in einer Datenbank.
- *displayName **
  ist der jederzeit veränderbare Anzeige-Name für die Gruppe
- *comment*
  Kommentar über Sinn und Zweck der Gruppe

2.2 Serverseitige Verwaltung von Benutzern und Gruppen

Zur serverseitigen Verwaltung von Benutzern und Gruppen dienen die im Folgenden beschriebenen Manager Klassen. Zu jeder Manager-Klasse gehört ein **EPI** (**E**sprit **P**rogramming **I**nterface), welches die vom Client aufrufbaren Methoden definiert.

➔ Alle Manager-Klassen unterliegen einer Berechtigungskontrolle. Ein Benutzer muss also die jeweils notwendigen Berechtigungen besitzen, um die EPI Methoden aufrufen zu können. Ein „Administrator“ (Benutzer, bei dem das *isAdmin*-Flag gesetzt ist) unterliegt dieser Kontrolle nicht und darf stets alles.

2.2.1 Verwaltung von Benutzern

User-Objekte werden serverseitig vom ***UserManager*** verwaltet, der das *ServerUserEPI* implementiert. Die Default-Implementierung *DefaultUserManager* speichert die konfigurierten Benutzer in der Server-Konfigurationsdatei **srvUser.ndf**.

2.2.2 Verwaltung von Gruppen

Gruppen werden serverseitig vom ***GroupManager*** verwaltet, der das *ServerGroupEPI* implementiert. Die Default-Implementierung *DefaultGroupManager* speichert die konfigurierten Gruppen in der Server-Konfigurationsdatei **srvGroup.ndf**. Der folgende Auszug zeigt die

Definition der Standardgruppen im Esprit-Server.

srvGroup.ndf

```
@TABLE Groups {
      [gid groupName        displayName       comment                    ]
       0   "administrators" "Administratoren" "Has all permissions"      ;
       1   "employees"      "Mitarbeiter"     "Has normal permissions"   ;
       2   "guests"         "Gäste"           "Has no permissions at all";
}
```

2.2.3 Verwaltung der Gruppenmitgliedschaft

Ein Benutzer kann grundsätzlich Mitglied von beliebig vielen Gruppen sein, oder von gar keiner. Die Gruppenzugehörigkeit spielt insbesondere dann eine wichtige Rolle, wenn es darum geht die Berechtigung eines Benutzers festzustellen. Sollte ein Benutzer ein bestimmtes Recht nicht explizit besitzen, wird ihm die betreffende Aktion normalerweise verwehrt. Ist er aber Mitglied einer Gruppe, die das Recht besitzt, dann wird ihm die Aktion auf diesem Wege erlaubt.

Die Gruppen-Mitgliedschaft wird serverseitig über die API des ***GroupMembersManagers*** verwaltet. Die Default-Implementierung *DefaultGroupMemberManager* speichert die Zuweisungen in der Server-Konfigurationsdatei **srvMember.ndf**. Der folgende Auszug zeigt einige beispielhafte Einträge:

srvMember.ndf

```
@LIST administrators {
       "admin"
       "tomcat"
}
@LIST employees {
       "rainer"
}
```

➔ Dem *GroupMemberManager* sind zur Laufzeit alle existierenden Benutzer und Gruppen bekannt. Er kann daher fehlerhafte Einträge in der Konfigurationsdatei (z.B. nicht mehr vorhandene Benutzer oder Gruppen) erkennen und automatisch korrigieren.

2.3 Benutzeroberfläche

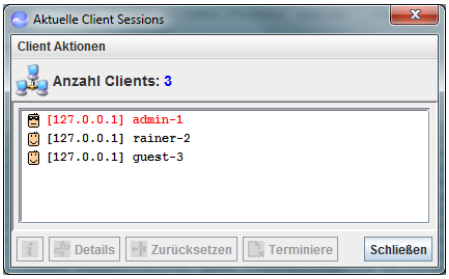
Im Toolbar des Administrations-Clients befinden sich die Werkzeuge zum Editieren von Benutzern und Gruppen, sowie zur Vergabe von Berechtigungen:

- **Clients**
  Öffnet den Dialog zur Verwaltung von Client-Sessions.
- **Benutzer**
  Öffnet den Dialog zur Verwaltung von Benutzern.
- **Gruppen**
  Öffnet den Dialog zur Verwaltung von Gruppen.
- **Rechte**
  Öffnet den Dialog zur Verwaltung von Benutzer- und Gruppenrechten für Server-Dienste, die einer Berechtigungskontrolle unterliegen.
- **Pfadrechte**

Öffnet den Dialog zur Verwaltung von Benutzer- und Gruppenrechten für serverseitige Filesystem-Pfade.

Sämtliche Aktionen dieser Werkzeuge sind in „Esprit-Manier“ implementiert, d.h. alle getätigten Änderungen werden den aktuell verbundenen Clients mitgeteilt und sind dort unmittelbar sichtbar.

2.3.1 Verwaltung von Sessions



Das Bild zeigt den *SessionManagerDialog,* der alle aktuell mit dem Server verbundenen Clients anzeigt. Die eigene Session ist rot hervorgehoben. Der Administrator kann Informationen von einer Session abfragen. Er kann sie re-initialisieren oder sie gar zwangsweise terminieren. In letzterem Fall erhält der Benutzer eine entsprechende Nachricht, dass die Verbindung zum Server getrennt wurde.

2.3.2 Verwaltung von Benutzern

Das Bild zeigt den *UserManagerDialog* zum Editieren von Benutzern. Der aktuelle Benutzer, als der man gerade selbst arbeitet, ist rot hervorgehoben. Diesen Benutzer kann man selbstverständlich nicht löschen.

Über die unten stehenden Aktionsknöpfe „Neu“ und „Ändern“ öffnen jeweils einen Editier-Dialog zum Neuanlegen bzw. Ändern eines Benutzers. „Löschen“ entfernt einen Benutzer vollständig. Dabei werden auch alle Gruppenzugehörigkeiten, sowie alle Berechtigungen des Benutzers gelöscht.

2.3.3 Verwaltung von Gruppen

Das Bild zeigt den *GroupManagerDialog* zum Editieren von Gruppen.

Die Liste links zeigt alle existierenden Gruppen. Die Liste auf der rechten Seite zeigt sämtliche Mitglieder der gerade selektierten Gruppe.

Die unten stehenden Aktionsknöpfe „Neu“ und „Ändern“ öffnen jeweils einen Editier-Dialog zum Neuanlegen bzw. Ändern einer Gruppe. „Löschen“ entfernt die Gruppe vollständig. Dabei werden auch alle Berechtigungen der Gruppe gelöscht.

➔ Eine Gruppe kann nicht gelöscht werden, solange ihr noch Benutzer zugewiesen sind.

2.3.4 Verwaltung der Gruppen-Mitgliedschaft

Die Funktionalität zum Ändern der Gruppen-Mitgliedschaft ist im *UserManagerDialog* integriert. Der Aktionsknopf „Gruppen“ öffnet den rechts gezeigten *GroupAssignDialog*, in dem die aktuellen Gruppenmitgliedschaften markiert sind und per Mausklick geändert werden können.

3 Das Permission-System

Der Esprit-Server beinhaltet ein Permission-System,welches die Zugriffe von Clients auf Server-Komponenten kontrolliert. Viele Server-Komponenten unterliegen einer Zugriffskontrolle. Für bestimmte Aktionen auf solchen Komponenten muss ein Client die dafür benötigte Berechtigung besitzen, damit er die Funktionalität nutzen kann. **Ein Administrator-Benutzer besitzt automatisch stets alle Rechte.** Mit Hilfe des *PermissionManagerDialogs* können Client-Rechte vom Administrator konfiguriert werden.

3.1 Schnittstellen des Permission-Systems

Das Permission-System des Esprit-Servers ist vollständig durch Schnittstellen (Interfaces) definiert und daher grundsätzlich austauschbar. Selbstverständlich existieren von allen Interfaces Default-Implementierungen, mit denen der Server standardmäßig betrieben wird. Die Schnittstellen sind:

- **EspritPermission**
  definiert eine Berechtigung, die einem Benutzer oder einer Gruppe zugesprochen werden kann. Eine solche Berechtigung ist typischerweise eine Enum-Konstante, die das Interface *EspritPermission* implementiert.
- **PermissionControlled**
  befähigt ein Server-Objekt zur Teilnahme an der Berechtigungskontrolle.
- **PermissionDefinition**
  definiert die unterstützten Berechtigungen eines Server-Objekts als eine Liste von *EspritPermissions*.
- **PermissionManager**
  verwaltet serverseitig die Berechtigungen von Benutzern und Gruppen.

3.2 Berechtigungen serverseitig

3.2.1 Server-Objekte mit Berechtigungskontrolle

Der Server besitzt eine ganze Reihe von Komponenten, deren Nutzung an Berechtigungen gebunden ist – sie alle implementieren das Interface *PermissionControlled* und können deshalb nach einem *PermissionDefinition* Objekt befragt werden. Letzteres beschreibt, welche Rechte jeweils unterstützt werden. Über den *PermissionManager* sind alle *PermissionControlled-Objekte* des Servers abrufbar und können dem Client in einem Dialog angezeigt werden.

3.2.2 Zuweisung von Berechtigungen

Die Vergabe einer Berechtigung stellt eine Verbindung zwischen folgenden drei Komponenten her:

- Server-Objekt (PermissionControlled)
- Berechtigung (EspritPermission - entspricht einer erlaubten Aktion)
- Berechtigter (User oder Group)

Der *PermissionManager* ist die zentrale Instanz im Server, die derartige Zuweisungen verwaltet und bei Zugriffen auf Server-Objekte prüft, ob die gewünschte Aktion für den gerade zugreifenden Benutzer erlaubt ist. Bei nicht erlaubtem Zugriff wird eine entsprechende *PermissionException* geworfen und so die jeweilige Aktion unterbunden.

Die Default-Implementierung *DefaultPermissionManager* speichert die konfigurierten Berechtigungen in der Server-Konfigurationsdatei **srvPermission.ndf**. Der folgende Auszug zeigt

ein paar beispielhafte Einträge für User- und Group-Permissions.

srvPermission.ndf

```
@TABLE GroupPermissions {
      [groupName   accessibleObject    permission]
       "employees" "LogChannelManager" ["READ" "CREATE"];
}
@TABLE UserPermissions {
      [userName accessibleObject   permission]
       "guest"  "SessionManager"   ["LIST_SESSIONS"];
       "rainer" "BatchTaskManager" ["EXECUTE" "CONFIGURE"];
}
```

3.2.3 Ermittlung einer Berechtigung

Basis für die Ermittlung einer Berechtigung ist die aktuelle *SessionId* eines Benutzers. Ist der Benutzer eingeloggt, dann enthält sie eine gültige *UserId*, die den Benutzer authentifiziert.

Es gilt das *default-denial* Prinzip: Ohne Authentifizierung (Login) und ohne Berechtigung geht nichts! Sollte für einen Benutzer die erforderliche Berechtigung für den Zugriff auf ein Server-Objekt explizit definiert sein, so darf er selbstverständlich die betreffende Aktion ausführen.

Sollte eine explizite Berechtigung nicht existieren, dann wird überprüft, ob die verlangte Berechtigung als Gruppen-Recht vorliegt. Dazu werden die Berechtigungen aller Gruppen geprüft, in der der betreffende Benutzer Mitglied ist. Nur, wenn die verlangte Berechtigung gefunden wurde, wird der Zugriff gewährt.

Bei folgenden Ausnahmen wird wird **keine** Berechtigungskontrolle durchgeführt:

- ➔ **Administrator-Benutzer**
  Dies ist ein Benutzer, der als Administrator definiert wurde. Er hat in seiner *SessionId* das *isAdmin*-Flag gesetzt und darf daher immer alles.
- ➔ **Server interne Zugriffe**
  Wenn der Server selbst Permission-geschützte Aufrufe macht, dann identifiziert er sich mit seiner *ServerSessionId* und erhält damit stets die Erlaubnis.
- ➔ **CoServer-Verbindungen**
  Server untereinander identifizieren sich mit einer *CoSessionId*. Diese laufen grundsätztlich ohne Authentifizierung. Damit haben Server untereinander praktisch immer Admin-Recht.

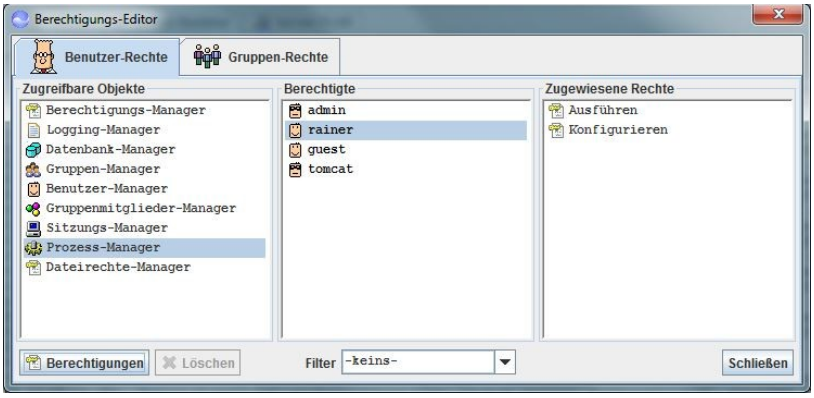
3.3 Benutzeroberfläche

3.3.1 Der PermissionManager-Dialog

Im Toolbar des Administrations-Clients befindet sich die Aktion zum Aufruf des *PermissionManagerDialogs*. Er zeigt drei Listen:

- Liste der zugreifbaren **Server-Objekte.**
  Dies sind die Server-Objekte, die unter Berechtigungskontrolle stehen (PermissionControlled).
- Liste der Berechtigten (alle existierenden **Benutzer** bzw. **Gruppen**).
  Der vorderste Reiter enthält in der mittleren Liste die Benutzer, der hintere Reiter die Gruppen. Ansonsten sind beiden die Reiter völlig analog.

- Liste der zugewiesenen **Rechte**.
  Dies können natürlich immer nur solche Rechte sein, die auch von dem betreffenden Server-Objekt unterstützt werden. Diese Liste kann nur dann einen Inhalt haben, wenn sowohl ein Server-Objekt, als auch ein Berechtigter selektiert ist.



Ein Klick auf den Aktionsknopf „Berechtigungen“ öffnet den Editor für Berechtigungsvergabe. Er zeigt alle für das gerade selektierte Server-Objekt definierten Berechtigungen an, wobei die bereits zugeteilten Berechtigungen markiert sind. Per Mausklick können weitere Berechtigungen für den selektierten Benutzer vergeben bzw. Vorhandene zurückgenommen werden.

Nach der Bestätigung sind die neu vergebenen Rechte sofort gültig. Alle anderen Clients werden darüber informiert. Ganz analog geschieht die Vergabe von Berechtigungen für Gruppen.

4 Das Pfadrechte System

Das Pfadrechte System dient der Zugriffskontrolle auf serverseitige Filesystem-Pfade. Der Server verwaltet ein bestimmtes Daten-Verzeichnis[2], auf welches er den Clients Zugriff gestattet. Dieser Zugriff ist allerdings Rechte-gebunden und nur für Clients erlaubt, für die die nötigen Rechte konfiguriert wurden.

Jeder Client kann eine lokale Sicht (*PathTree*) auf das Filesystem des Servers haben, sieht allerdings nur solche Verzeichnisse und Dateien, für die er entsprechende Pfadrechte besitzt. Ohne Rechte sieht er lediglich den leeren Wurzel-Knoten. Pfadrechte sind auf konkrete Pfade bezogen und können entweder benutzer- oder gruppenspezifisch vergeben werden. Ein Administrator-Benutzer hat – wie üblich – automatisch stets alle Rechte.

2 Das Daten-Verzeichnis kann in der Server-Startup Konfiguration mit Hilfe des Konfigurations-Schlüssels *esprit.server.watchedDataDir* definiert werden. Standardmäßig ist das Verzeichnis *<serverWorkspace>/dataDir* eingestellt.

4.1 Path-Watching

Das Daten-Verzeichnis des Servers unterliegt einer ständigen Beobachtung durch einen sog. *PathWatcher*. Jede Änderung im Filesystem wird registriert und an die Clients notifiziert, so dass die clientseitige Filesystem-Sicht stets aktuell ist. Dies gilt auch bei Änderungen der Pfadrechte Konfiguration. Clients sehen also Pfade kommen und verschwinden, je nachdem, wie ihre Rechte gerade vom Administrator konfiguriert wurden.

4.1.1 Path-Manager

Der *PathManager* ist die Server-Komponente, die verantwortlich ist für das Beobachten des Server-Filesystems, insbesondere des Daten-Verzeichnisses, auf das den Clients Zugriff gewährt wird. Er besitzt eine zentrale Verwaltung aller beobachten Verzeichnis-Pfade. Änderungen in der Pfad-Registrierung werden per *RemotePathRegistrationEvent* an die Clients notifiziert. Ein Client kann also eine stets aktuelle Sicht aller im Server registrierten Pfade haben.

4.2 Visualisierung des Filesystem-Baumes

Zur Visualisierung eines lokalen Filesystem-Baumes im Client dient die *PathTree* GUI-Komponente. Sie nutzt einen zugrundeliegenden *LocalPathTreeModelDriver* als Datenquelle und Schnittstelle zum Filesystem.

Für die Visualisierung eines remoten (auf dem Server befindlichen) Filesystems wird der Treiber allerdings gegen einen *RemotePathTreeModelDriver* ausgetauscht. Dieser delegiert alle Aufrufe über Netz an einen serverseitigen *ServerPathTreeModelDriver*, der seinerseits direkten Zugriff auf das Server-Filesystem hat. Letzterer ist eine Erweiterung eines *LocalPathTreeModelDrivers* und berücksichtigt bei der Auslieferung von Pfaden die jeweils vergebenen Benutzerrechte.

4.2.1 PathInfo und RemotePathInfo

Ein *LocalPathTreeModelDriver* beliefert die *PathTree* GUI-Komponente mit *PathInfo* Objekten, die entweder jeweils einen Datei- oder ein Verzeichnis-Knoten repräsentieren. Ein *RemotePathTreeModelDriver* liefert hingegen (vom *ServerPathTreeModelDriver* stammende) *RemotePathInfo* Objekte, die zusätzlich die Information beinhalten, welche Rechte der Benutzer auf dem betreffenden Pfad hat[3]. Pfade, auf denen der Benutzer keine Rechte hat, werden gar nicht erst ausgeliefert.

4.3 Verwaltung von Pfadrechten

Welcher Benutzer auf welchem Pfad welche Rechte besitzt, wird im *PathPermissionManager* persistent konfiguriert. Da *er* diese Information besitzt, ist *er* auch derjenige, der die *RemotePathInfo* Objekte erzeugt und mit Rechten versieht. Diese Rechte sind natürlich benutzer- bzw. gruppenspezifisch. Der gleiche Pfad kann für verschiedene Benutzer unterschiedliche Rechte besitzen. Selbstverständlich muss ein Benutzer authentifiziert (eingeloggt) sein, um überhaupt zugreifen zu können.

4.3.1 Caching von Treibern

Viele Benutzer können grundsätzlich gleichzeitig im gleichen Server-Filesystem arbeiten. Ihre Aktionen können dabei in in gleichen oder völlig verschiedenen Unterverzeichnissen stattfinden. Jeder Benutzer sieht dabei jeweils nur die Knoten, auf denen er die nötigen Rechte besitzt. Um diese „Durchmischung“ zu bewerkstelligen, wird beim jeweils ersten Zugriff eines Benutzers auf

3 Es wäre technisch sehr aufwendig und wenig performant, wenn die Rechte eines jeden Pfades vom Client erfragt werden müssten. Daher werden sie im *RemotePathInfo* Objekt gleich mitgeliefert.

das Server-Filesystem für diesen eine eigene Instanz eines *ServerPathTreeModelDrivers* erzeugt und solange im Cache gehalten, bis der Benutzer sich ausloggt.

4.3.2 Änderungen im Server-Filesystem

Alle Verzeichnis-Pfade des Server-Filesystems, die von Benutzern zugegriffen wurden, werden automatisch vom *PathTreeManager* zur künftigen Beobachtung registriert. Sollten - wodurch auch immer - dort neue Dateien/ Verzeichnisse erzeugt, geändert oder gelöscht werden, so wird dies unmittelbar erkannt und in Form von *RemotePathChangeEvents* an die Clients kommuniziert.

4.3.3 Änderungen von Pfadberechtigungen

Ein besonderer Fall tritt dann ein, wenn die Konfigurationsdatei des *PathPermissionManagers* (*srvPathPermission.ndf*) geändert wurde. Dann nämlich wurden Rechte von Pfaden geändert und persistiert. In diesem Fall wird dem Client ein *PathPermissionConfigChangeEvent* geschickt, woraufhin dieser seinen kompletten PathTree neu laden muss, da sich seine Rechte völlig geändert haben können. Der Server schickt dieses Event wenn möglich nur an solche Clients, die definitiv betroffen sind.

4.4 Pfadrechte-Manager

Der *PathPermissionManager* ist die zentrale serverseitige Instanz zur Verwaltung der vergebenen Pfad-Rechte (*PathPermissions*). Er persistiert diese in der Server-Konfigurationsdatei *srvPathPermission.ndf*. Pfad-Rechte können für Benutzer oder Gruppen vergeben werden. Besitzt ein Benutzer keine spezifischen Rechte, so gelten die Rechte der Gruppen, bei denen er Mitglied ist.

4.4.1 Unterstützte Pfad-Rechte

Welches die unterstützten Rechte für Pfade sind, wird bestimmt durch eine ***PathPermissionMaster*** Instanz im *PathPermissionManager.* Die Standard-Implementierung *DefaultPathPermissionMaster* unterstützt die folgenden Standard-Rechte.

- Standard Rechte für **Verzeichnisse**

READ,	Lesen der Einträge
MODIFY,	Einträge erzeugen/löschen
DOWNLOAD	Wird vererbt an alle Einträge

- Standard Rechte für **Dateien**

READ,	Lesen des Inhalts
MODIFY,	Ändern des Inhalts
DOWNLOAD	Herunterladen
EXECUTE	Ausführen der Datei (nur möglich falls dies ein Programm ist)

Hierbei ist das READ-Recht das jeweils Niedrigste. Alle anderen Rechte beinhalten dieses Recht implizit.

Der *PathPermissionMaster* kann durch eine kundenspezifische Implementierung ersetzt werden, die völlig andere Rechte definiert, wie z.B. ein EXTRACT Recht zum Entpacken von Zip-Dateien, oder ein ARCHIVE Recht zur Archivierung bestimmter Verzeichnisse. Jeder einzelne Pfad kann im Prinzip andere Rechte haben.

4.4.2 Vererbung von Pfad-Rechten

➔ **Aufwärts-Vererbung**
Gegeben sei ein Dateipfad */A/B/C/look.txt*. Damit ein Benutzer die Datei *look.txt* ändern darf, müsste er auf den Verzeichnissen A, B, und C, sowie auf der Datei selbst Leserechte haben. Dies zu konfigurieren, wäre für den Administrator sehr aufwändig, daher genügt es vielmehr, wenn die Datei *look.txt* alleine das Änderungsrecht bekommt. Es enthält impizit das Leserecht für alle darüberliegenden Verzeichnisse, obwohl diese selbst keine Rechte besitzen.
Also: bei der Vergabe eines beliebigen Rechtes für einen Pfad vererbt sich das jeweils niedrigste Recht (hier das Leserecht) an alle darüberliegenden Pfade.

➔ **Abwärts-Vererbung**
Gegeben sei ein wiederum der Dateipfad */A/B/C/look.txt*. Wird dem Verzeichnis-Knoten *A* das Änderungsrecht vergeben so vererbt sich dieses auf alle Unterverzeichnisse (hier *B* und *C*). Auch die Datei- *look.txt* erbt dieses und darf dann vom Client verändert werden.
Also: bei der Vergabe eines bestimmten Rechtes für einen Pfad vererbt sich dieses an alle darunterliegenden Pfade.

Ob ein Client einen Pfad aufgrund seiner Rechte sehen darf oder nicht, entscheidet technisch die *PathPermissionFilter* Komponente. Sie macht alle Pfade ohne Rechte für den Client unsichtbar. Diese Komponente ist austauschbar. Eine kundenspezifische Implementierung könnte zusätzlich bestimmte Pfade für den Client gänzlich ausblenden.

4.4.3 Löschen von Pfaden

Wird ein Pfad, der mit Pfadrechten versehen ist, im Filesystem gelöscht, dann wird dadurch seine Rechte-Definition natürlich ungültig. Die *PathPermissionManager*-Komponente reagiert automatisch auf das Löschen von Pfaden und bereinigt selbständig – falls erforderlich – die Konfigurationsdatei für Pfad-Rechte.

4.4.4 Der Wurzel-Pfad (Root-Path)

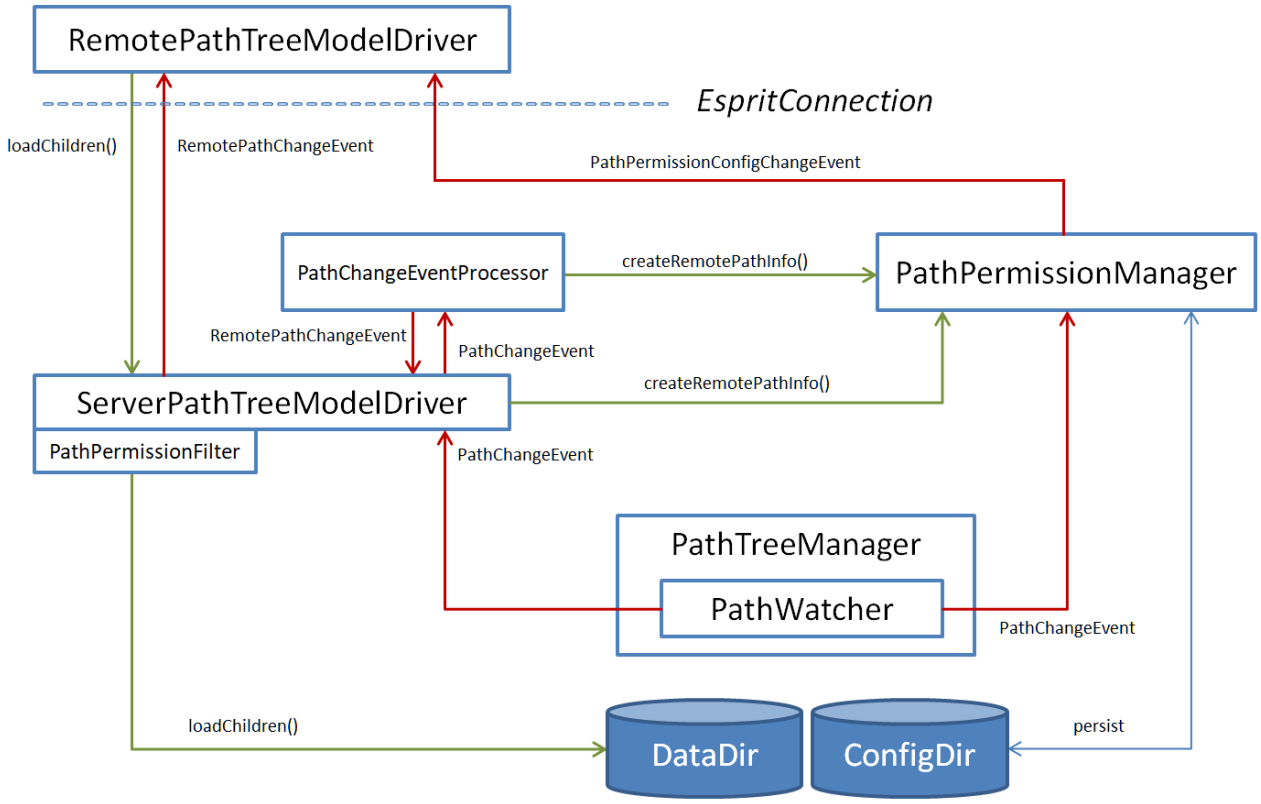
Der Esprit-Server stellt standardmäßig ein bestimmtes (konfigurierbares) Daten-Verzeichnis als Wurzel-Verzeichnis für Client-Zugriffe zur Verfügung. Dieses unterliegt der Beobachtung durch den Watch-Service des *PathTreeManagers.*

Dieser Wurzelpfad wird in der clientseitigen Filesystem-Sicht nicht als Pfadname, sondern als *Servername* dargestellt. Der Anwender kennt also im Prinzip nicht die wirkliche physikalische Lage des Wurzelpfades auf dem Server. Sämtliche Pfade unterhalb des Wurzelpfades erscheinen dem Client als relative Pfade.

Beim clientseitigen Zugriff auf ein Unterverzeichnis (Expansion eines Knotens in der Filesystem-Sicht) wird dieses automatisch im Watch-Service des *PathTreeManagers registriert* und unterliegt somit ebenfalls künftiger Beobachtung.

4.5 Übersicht über das Pfadrechte System

Das folgende Diagramm gibt eine technische Übersicht über die involvierten Komponenten und ihre Beziehungen zueinander. Es ist sehr nützlich, um den Source-Code zu verstehen. Für eine detaillierte Beschreibung sei auf die zugehörige JavaDoc-Dokumentation verwiesen.



4.6 Schnittstellen des Pfadrechte Systems

Das gesamte *Pfadrechte* System ist definiert über Schnittstellen (Interfaces) und daher im Prinzip austauschbar. Selbstverständlich existieren von allen Interfaces Default-Implementierungen, mit denen der Server standardmäßig betrieben wird. Grundsätzlich kann das System ganz oder teilweise durch eine kundenspezifische Implementierung ersetzt werden, die beispielsweise sämtliche Konfigurationsdaten in einer Datenbank persistiert.

Eine vollständige Beschreibung aller Schnittstellen und Basisklassen des Pfadrechte Systems befindet sich in der zugehörigen JavaDoc-Dokumentation.

4.7 PathPermissionSets

Es wäre aufwendig und unübersichtlich für den Administrator, wenn er viele Pfade mit Einzelpermissions versehen müsste. Um dies zu vereinfachen, gibt es die *PathPermissionSet* Klasse, die mehrere *PathPermissions* gruppiert. Ein solcher *PathPermissionSet* kann einem Pfad im Projekt-Baum für einen Benutzer oder eine Gruppe zugewiesen werden. Damit erhält der betreffende Benutzer bzw. die betreffende Gruppenmitglieder alle im Set enthaltenen *PathPermissions* auf dem betreffenden Pfad.

4.7.1 Serverseitige Implementierung

Im Esprit-Server ist ein *ServerPathPermissionSetManager* eingebaut, der über eine *ServerPathPermissionEPI* das Erzeugen, Ändern und Löschen von *PermissionSets* erlaubt. Sämtliche Änderungen werden unmittelbar in der Server-Konfigurationsdatei **srvPermissionSet.ndf** persistiert und es wird jeweils ein *PermissionSetChangeEvent* an alle Admin-Clients geschickt, so dass diese sich aktualisieren können.

Beispiel für eine Server-Konfigurationsdatei: **srvPermissionSet.ndf**

```
<
      ndfDocVersion = "1.6"
      ndfDocEncoding = "UTF-8"
      ndfDocCreated = "2017-11-10 09:53:33"
      ndfDocType = "PermissionSets"
>
@OBJECT PermissionSets {
      @OBJECT PermissionSet {
            < permSetName="Berarbeiter" >
            @LIST permissions {
                  "ARCHIVE_DIR"
                  "EXECUTE_FILE"
                  "PATH_DOWNLOAD"
                  "PATH_MODIFY"
                  "PATH_READ"
            }
      }
      @OBJECT PermissionSet {
            < permSetName="Auswerter" >
            @LIST permissions {
                  "PATH_MODIFY"
                  "PATH_READ"
            }
      }
}
```

4.7.2 Zuordnung von Strings zu PathPermissions bzw. PermissionSets

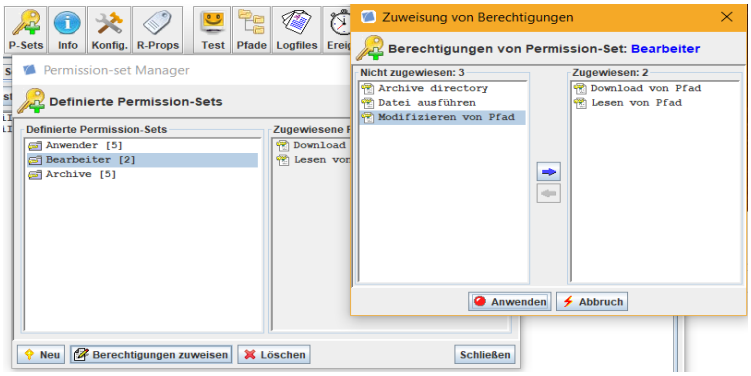
Für seine Funktion benötigt der *ServerPathPermissionManager* drei Instanzen, die ihm bei der Zuordung von Strings zu *PathPermissions* bzw. *PermissionSets* helfen:

- ➔ Der **PermissionResolver**
  Von einer *PathPermission* wird lediglich ihr eindeutiger Name persistiert. Um beim Einlesen der Konfigurationsdatei den Namen in eine *PathPermission* zurückzuwandeln muss im *ServerPathPermissionManager* ein ***PermissionResolver*** installiert sein, der alle verfügbaren Permissions kennt. Er kann kundenspezifisch gesetzt werden.
- ➔ Der **PermissionSetResolver**
  Im *ServerPathPermissionManager* muss darüberhinaus eine ***PermissionSetResolver*** Instanz installiert sein, die in der Lage ist, einen bestimmten *PermissionSet* anhand des Namens zu identifizieren. Der *ServerPermissionSetManager*, da er ja die in der Server-Konfigurationsdatei gespeicherten *PermissionSets* verwaltet, implementiert dieses Interface und wird bei der Server-Initialisierung dem *ServerPathPermissionManager* bekannt gemacht.
- ➔ Der **PathPermissionMaster**
  Die einem bestimmten Pfad zuweisbaren *PathPermissions* werden von einem *PathPermissionMaster* definiert. Er sorgt dafür, dass einem Directory nur *DirectoryPermissions* und einer Datei nur *FilePermissions* zugewiesen werden können. Er kann mit kundenspezifischen *PathPermissions* befüllt werden.

4.7.3 Clientseitige Implementierung

Im Esprit-Client wurde ein entsprechender *PermissionSetManagerDialog* eingebaut, mit dessen Hilfe ein Administrator-Benutzer *PermissionSets* erzeugen, editieren und löschen kann.

Für das Erzeugen eines *PermissionSets* muss ein eindeutiger Name vergeben werden. Mit Hilfe des unten im Bild gezeigten Listen-Dialogs können sodann die gewünschten *PathPermissions* aus einer Menge von Verfügbaren ausgewählt und zugewiesen werden.



PermissionSets können jederzeit vom Administrator geändert werden, selbst wenn sie aktuell bei laufenden Clients in Benutzung sind. Bei einer Änderung werden alle Admin-Clients mit Hilfe eines *PermissionSetChangeEvents* informiert und aktualisieren jeweils ihre Sicht.

Für einen normalen Client, der die Sicht auf einen auf den Projekbaum offen hat, können sich bei einer Veränderung eines *PermissionSets* die Berechtigungen völlig verändern. Er wird bei sochen Änderungen durch ein *PathPermissionConfigChangeEvent* aktualisiert. Je nach Benutzerrechten verschwinden dann in seinem Projektbaum bestimmte Pfade bzw. es werden Neue sichtbar.

Der Projektbaum des Clients reagiert auf zwei Events:

➔ **PathChangeEvent**
Wird geschickt, wann immer sich ein Pfad im Filesystem ändert. Dabei erhält ein Client Events nur von Pfaden, die seinen Projektbaum betreffen.

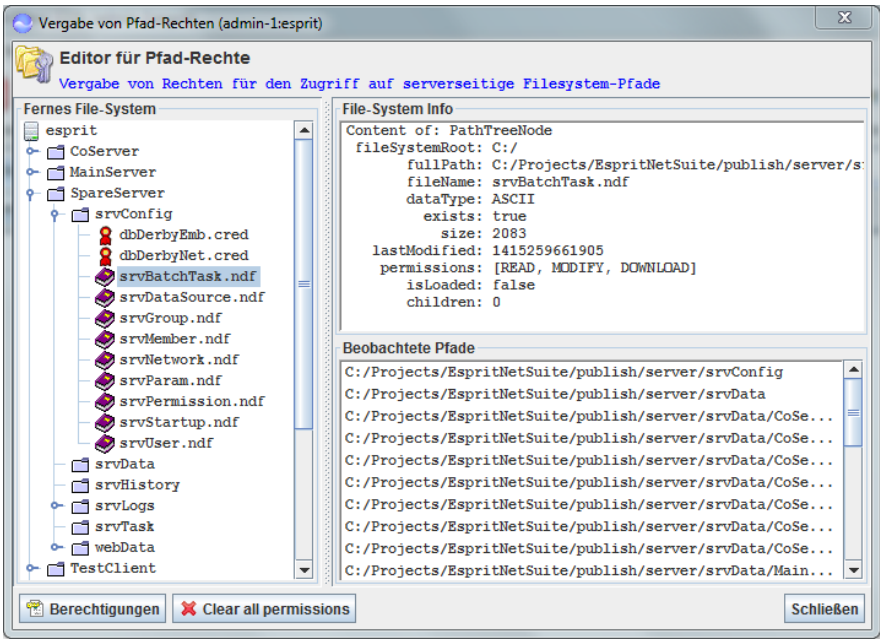
➔ **PathPermissionConfigChangeEvent**
Wird geschickt, wann immer sich die Pfadrechte eines Clients geändert haben könnten. Der Server informiert, wenn möglich, nur soche Clients, die auch betroffen sind.

4.8 Benutzeroberfläche

4.8.1 Der PathPermissionManager-Dialog

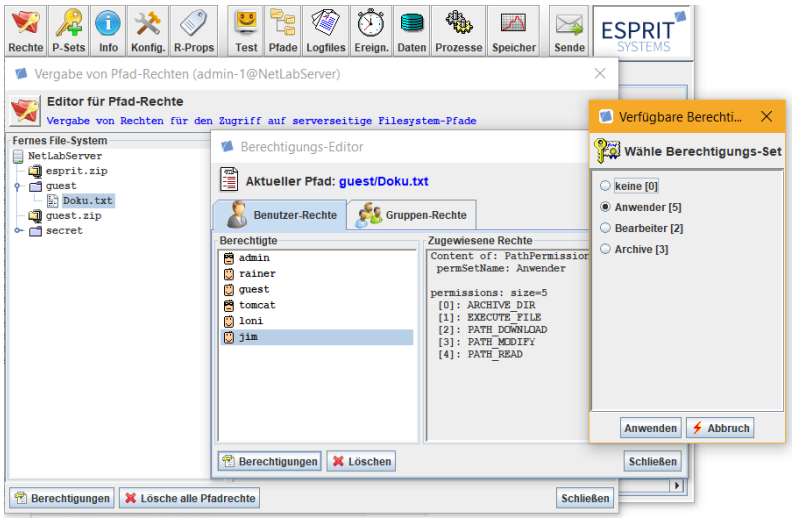
Im Toolbar des Administrations-Clients befindet sich die Schaltfläche zum Aufruf des *PathPermissionManagerDialogs,* mit dessen Hilfe Pfad-Rechte vergeben werden können. Er zeigt drei Bereiche:

- **Datei-Baum**
  Dies ist eine Visualisierung des serverseitigen Daten-Verzeichnisses.
- **Datei/Verzeichnis Info-Fenster**
  Hier wird Detailinformation zum jeweils selektierten Pfad angezeigt.
- **Pfad-Registrierung**
  Zeigt eine Liste aller serverseitig registrierten Pfade, die unter Beobachtung stehen.



Das Anklicken des „*Berechtigungen*“ Knopfes unten links öffnet den Editor zur Verwaltung der Rechte auf dem gerade selektierten Pfad. Der Editor zeigt links eine Liste aller möglichen Berechtigten (Benutzer bzw. Gruppen). Für den selektierten Berechtigten werden im rechts gezeigten Popup-Dialog die definierten *PermissionSets* zur Auswahl angeboten. Nach Bestätigung der Auswahl werden die effektiv zugewiesenen Rechte im Sichtfenster des Editors angezeigt. Die Zuweisung eines *PathPermissionSets* kann jederzeit geändert oder gelöscht werden.

➔ Man beachte, dass eine Änderung der Pfadrechte auf allen Clients sofort wirksam ist! Ein Client kann also Pfade verschwinden sehen, auf denen er die Rechte entzogen bekommen hat. Umgekehrt erscheinen ggf. neue Pfade, auf denen er Rechte bekommen hat.

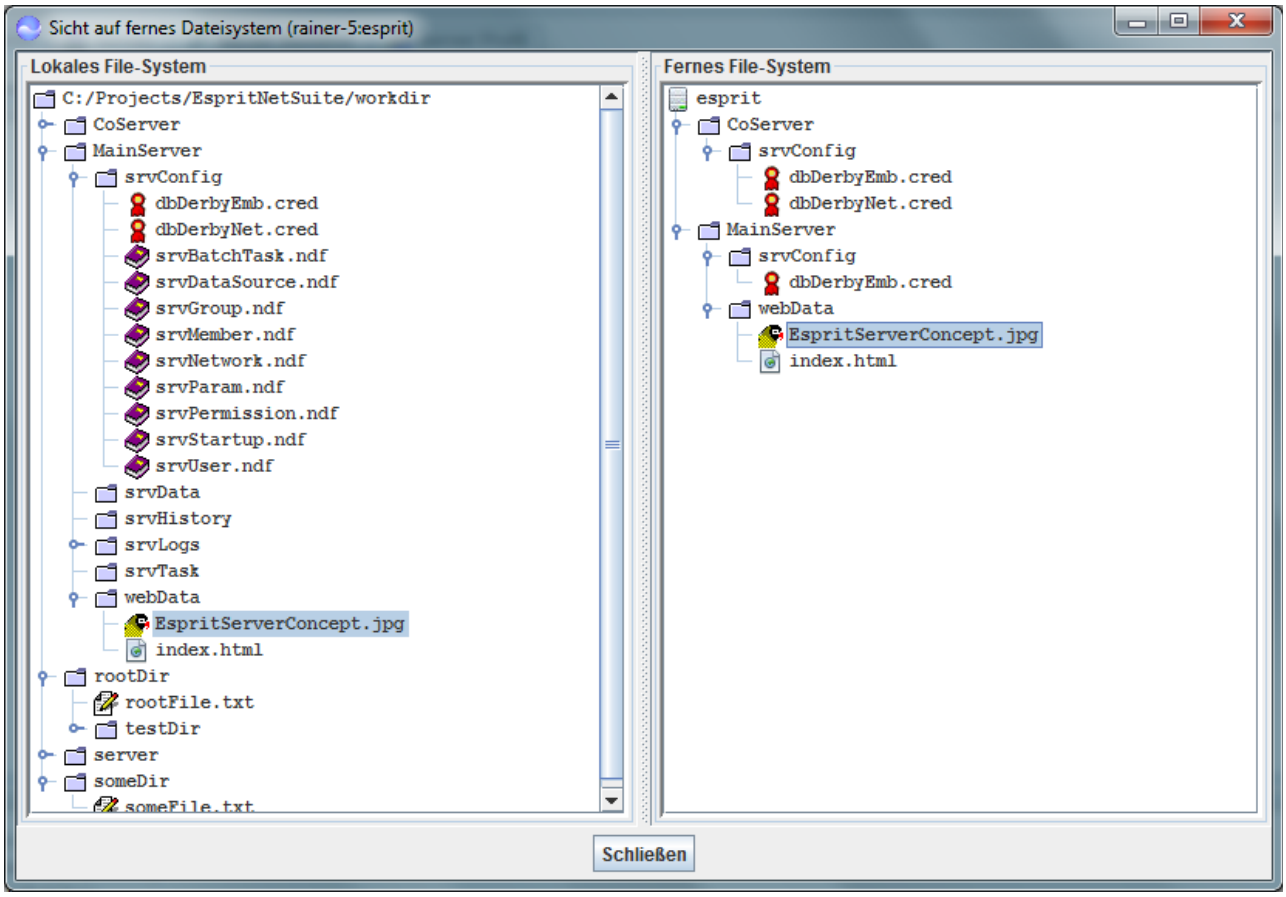


4.8.2 Der RemotePathTree-Dialog

Das *RemotePathTreeTool* ist ein Client-Werkzeug zum bequemen Austausch von Dateien zwischen dem lokalen Filesystem (im Bild links) und dem Server-Filesystem (im Bild rechts).

Die Wurzel des lokalen Filesystems ist das (konfigurierbare) Arbeitsverzeichnis des Clients; die Wurzel des Server-Filesystems ist typischerweise ein speziell konfiguriertes Datenverzeichnis,

dessen absoluter Pfad für den Client unsichtbar bleibt. Beide Filesysteme unterliegen der Überwachung durch einen Watch-Service, so dass alle Filesystem-Änderungen in beiden Ansichten sofort automatisch aktualisiert werden.



4.8.3 Standard-Aktionen in den Filesystem-Ansichten

Beide Filesystem-Ansichten besitzen ein Kontext-Menü mit diversen Standard-Aktionen. Je nachdem, welcher Pfad gerade selektiert ist, werden die dort jeweils möglichen Aktionen angezeigt. Aktionen im Server-Filesystem erfordern bestimmte Pfadrechte. Ohne Rechte werden sie entweder gar nicht angeboten oder aber sie scheitern, wenn die erforderlichen Rechte geprüft werden und nicht vorhanden sind. Die implementierten Standard-Aktionen sind:

- **Baum neu laden**
  Lädt den kompletten Baum neu vom Filesystem.
- **Knoten neu laden**
  Lädt nur den selektierten Pfad neu vom Filesystem.
- **Alles expandieren/kollabieren**
  Expandiert/kollabiert einen Pfad rekursiv komplett bis zu den Blattknoten.

➔ Achtung: die Expansion des Wurzel-Knotens führt dazu, dass das gesamte Filesystem eingelesen wird – dies kann bei einem großen Filesystem erhebliche Zeit dauern. Daher wird der Anwender hier um Bestätigung gebeten. Das Kollabieren eines Knotens ist hingegen eine sehr schnelle Operation.

- **Erzeugen/Umbenennen/Löschen von Verzeichnis/Datei**
  Erzeugt ein Verzeichnis bzw. eine Datei.
  Benötigt *MODIFY*-Berechtigung auf dem darüberliegenden Verzeichnis.

➔ **Archivieren (ZIP) eines Verzeichnisses**
Startet einen Workflow, der ein ein ZIP-Archiv des selektierten Verzeichnisses unter dem Namen *<directoryName>.zip* erzeugt. Danach kann das Verzeichnis selbst im Prinzip gelöscht werden, da es aus dem ZIP-Archiv jederzeit wiedergewonnen werden kann.
Benötigt *MODIFY*-Recht auf dem darüberliegenden Verzeichnis.

➔ **Auflistung eines Archivs**
Startet einen Workflow, der den Inhalt des Archivs (ZIP oder TAR) anzeigt.
Benötigt *READ*-Recht auf der Archiv-Datei.

➔ **Extraktion eines Archivs**
Startet einen Workflow, der das Archiv (ZIP oder TAR) extrahiert. Dabei werden i.d.R. neue Verzeichnisse und Dateien angelegt.
Benötigt *MODIFY*-Recht auf dem darüberliegenden Verzeichnis.

➔ **Textdatei einsehen** (nur im Serverfilesystem)
Startet das *TextViewer*-Werkzeug und zeigt den Inhalt der selektierten Textdatei.
Achtung: Aus Performancegründen ist der Inhalt begrenzt auf einige Tausend Zeilen.
Zum Editieren muss die Datei heruntergeladen, lokal geändert, und wieder hochgeladen werden.
Benötigt *READ*-Recht auf der Datei.

➔ **Datei öffnen** (nur im Clientfilesystem)
Öffnet die selektierte Datei mit dem im Betriebssystem definierten Werkzeug. Hierbei wird die Desktop-Integration des jeweiligen Betriebssystems verwendet.

➔ **Pfad suchen** in der jeweils anderen Filesystemansicht
Sucht den gerade selektierten Pfad in der gegenüberliegenden Filesystemansicht und selektiert ihn, falls er dort gefunden wurde.

Die folgenden Aktionen dienen dem Austausch von Dateien zwischen Client und Server. Hierbei werden die Daten per Esprit-Filetransfer übertragen. Eine übertragene Datei wird auf der Zielseite im gleichen relativen Pfad erscheinen wie auf der Quellseite. Die ggf. dazu erforderlichen Unterverzeichnisse werden automatisch erstellt.

➔ Client- und Server-Filesystem behalten also stets die gleiche Struktur! Sie kann durch Up/Downloads nicht verletzt werden.

➔ **Datei-Upload** (nur Clientseite)
Lädt die selektierte Datei hoch zum Server.
Benötigt *MODIFY*-Recht auf dem serverseitigen Zielverzeichnis .

➔ **Datei-Download** (nur Serverseite)
Lädt die selektierte Datei herunter zum Client.
Benötigt *DOWNLOAD*-Recht auf der Datei oder dem darüberliegenden Verzeichnis.

Das Hoch-/Herunterladen von Verzeichnissen kann leicht über einen Umweg erreicht werden: Man archiviert das betreffende Verzeichnis in eine lokale ZIP-Datei, transerferiert diese und entpackt sie wieder am Zielort. Zum Hochladen des Zip-Archivs ist selbstverständlich das *MODIFY*-Recht auf dem darüberliegenden Verzeichnis erforderlich.