Raid-Rush: Loader Spezifikation

Revision 0.1a

27. Mai, 2012

Überblick

Was kommt tatsächlich in den Loader und der GUI hinein und wie sind die Komponenten aufgebaut? Hier einmal ein paar Fakten über die einzelnen Bestandteile.

Loader

(inkl. Encrypter)

Kern

Im Kern gibt es einen internen Server, welcher sich nur damit beschäftigt über ein internes Protokoll mit den UIs zu kommunizieren. Wie genau das Protokoll aufgebaut ist, wird auf folgenden Seiten genauer erläutert.

Der Kern bringt folgende Eigenschaften mit sich:

- **Ordnerverwaltung**
    - Containergeneration
    - Mappen eines Ordners an eine Kategorie
        - Kategoriezuweisung bei Containereinbindung
            - Nur wenn mehrere Kategorien vorhanden
- **Container**
    - Formate
        - **SFT**
        - **RRLC**
    - Verschlüsselung (AES, ..)
    - Priorität (Container a vor Container b)
- **Plugin-System**
- **Netzwerk**
    - **Multi-Threading**
        - Einstellbar. Limitierungen durch den Containerersteller
    - **Multi-Segment**
    - Protokolle
        - **HTTP**
        - **FTP**
            - Autom. Active/Passive Umschaltung
    - **SSL Unterstützung**
    - **Proxy** (Socks4, Socks5, ..)
    - **IPv6** Unterstützung
- **Cross Platform**
- Server mit **internem Protokoll**
    - Informationsaustausch zwischen UI und Loader
- **Multilingual**
- **CLI**
    - **ncurses**

Mehr folgend.

Plugin-System

Der Loader soll ein eigenes Plugin-System bekommen, welches zusätzliche Aktionen, wo auch immer sie benötigt werden, durch einfache Anbindungen an den Loader möglich werden. Darunter fallen für Loader typ. Aktionen wie ein *whois* oder wie das Entpacken der Archive.

Ablaufen sollte das folgendermaßen:

Es gibt ein Unterverzeichnis **/plugins**, welches die Plugins beinhaltet. In der

Konfigurationsdatei des Loaders stehen neben den anderen Informationen auch die zu ladenden Plugins für ein entsprechendes Rechtegebiet (Net, Container, etc. und die benötigten Funktionen, um die Abfragen zu veringern, wo die Plugins sowieso nicht gebraucht werden) mit vollständigem oder relativem Pfad. Geladen werden die Plugins an entsprechenden Punkten im Programm mit Daten gefüttert oder eben geholt. Das ganze läuft Asynchron und nebenher, um den Loader nicht an seinen Aufgaben zu hindern bzw. ihn zu blockieren.

Wie der Informationsaustausch hier stattfindet muss noch geklärt werden.

Folgende Plugins sind geplant:

- Clipboard Überwachung
- OCHs (u.A. XUP)
- Whois
- Unrar
- Checksummen (CRC, MD5, ..)
- Aufräumungen (Container löschen, div. Dateien löschen [z.B. *.nfo])
- Bietemaker / Speedreport (Nicht in naher Zukunft, da erstmal unnötig)
- Click and Load (Nicht in naher Zukunft, da erstmal unnötig)
- Dateisystem-Check (FS-Limitierungen bezügl. Datei-/Ordnernamen)
- URL Protocol Handler (Nicht in naher Zukunft, da erstmal unnötig
    - rrlc://user:pw@urlaubsfotos.rrl
- Auto-Tagger (Nicht in naher Zukunft, da erstmal unnötig)
    - mit regulären Ausdrücken
- Auto-Update (Nicht in naher Zukunft, da erstmal unnötig und die Mittel fehlen)

Mehr folgend.

User Interface

Web

Mehr folgend.

System

Mehr folgend.

Protokoll

Loader Protocol Version 0.1

Informationen über das RRL Protokoll. Das interne Protokoll dient dem Datenaustausch zwischen dem Loader und der UI um Informationen abzurufen oder den Loader zu kontrollieren.

Die unten aufgelisteten Strukturen sind nicht vollständig und nur Vorabversionen. Der tatsächliche Aufbau folgt.

Das, öfter aufgeführte, Feld „**Inhalt**“ enthält Informationen in JSON. Ein Protokoll in JSON wäre für weitere Nutzung denkbar (Javascript [Websockets] ist es unmöglich Pakete binär zu analysieren und interpretieren).

Momentan sind Informations- und Kontrollanfragen identisch, das könnte sich in Zukunft jedoch ändern und daher bleiben beide Arten der Anfragen vorhanden.

Genereller Protokollheader

Protocol Header			
Feld	**Länge (Byte)**	**Offset (Byte)**	**Beschreibung**
Version	1	0	Versionsnummer des Protokolls
Checksum	1	1	Checksumme der gesamten Struktur, die ausmacht: Header + Anfrage
Signature	4	2	Signatur der entsp. Anfrage
Identifikationsnummer	2	6	Dient der Identifikation für den Clienten, um mehrere Anfragen gleichzeitig zu bearbeiten und eindeutige Antworten liefern zu können.
Länge (Bytes)	4	8	Länge der Anfrage

Versionen

Version	Beschreibung
1	**Erste Version des Loaders**

Signaturen

Signatur	Beschreibung
„LIRQ“	**L**oader **I**nformation **Req**uest
„LCRQ“	**L**oader **C**ontro**l** **Req**uest
„LRQR“	**L**oader **Req**uest **R**esponse

Modulcodes

Code	Beschreibung
0	**Kein Modul**
1	**Container**
2	**Netzwerk**
3	**Plugin**
4	**Queue**

Informationen

Informationsanfrage			
Feld	**Länge (Byte)**	**Offset (Byte)**	**Beschreibung**
Header	12	0	Protokoll Header
Modul	1	12	Angesprochenes Modul.
Anfrage	1	13	Anfrage an das Modul
Inhalt	***n***	14	***JSON*** *Code*

Kontrolle

Kontrollanfrage			
Feld	**Länge (Byte)**	**Offset (Byte)**	**Beschreibung**
Header	12	0	Protokoll Header
Modul	1	12	Angesprochenes Modul.
Anfrage	1	13	Anfrage an das Modul
Inhalt	***n***	14	***JSON*** *Code*

Rückmeldung / Antwort

Rückmeldung			
Feld	**Länge (Byte)**	**Offset (Byte)**	**Beschreibung**
Header	12	0	Protokoll Header
Rückgabewert	1	10	Rückgabewert des jew. Requests
Inhalt	***n***	11	***JSON*** *Code*

Anfragecodes

Noch anzupassen

Modul: „Container“	
Code	**Beschreibung**
Modul: „Netzwerk“	
Code	**Beschreibung**
Modul: „Plugin“	
Code	**Beschreibung**
Allgemein	
Code	**Beschreibung**