ELSTER-*Transfer*

Fortgeschrittene Konfiguration

ETR Version 26.01

Dokumentversion: 27

Veröffentlicht am 27.01.2026

Herausgeber: Bayerisches Landesamt für Steuern

Inhaltsverzeichnis

1 Serverbetrieb (Sicherheitsmaßnahmen bei Remote-Zugriff)

ELSTER-Transfer bietet eine REST-Schnittstelle (REST-API) an, um von anderen Rechnern aus ETR-Funktionen automatisiert aufzurufen bzw. Informationen zu Datenübertragungen abfragen zu können.

Damit erweitert sich das bisherige Einsatzgebiet der Anwendung (Clientanwendung auf Desktop-Rechnern) um die Möglichkeit, die Anwendung im Servermodus zu betreiben (Remote-Zugriff). Dieser muss explizit freigeschaltet werden (→ Remote-Zugriff aktivieren und konfigurieren).

Gleichzeitig entstehen bei Serverbetrieb bzw. aktiviertem Remote-Zugriff durch die erhöhten Anforderungen aber auch neue Risiken, die besondere Maßnahmen für den sicheren Betrieb der Anwendung erfordern. Diese sind in den folgenden Abschnitten beschrieben.

Fortgeschrittene Konfiguration - Wichtiger Hinweis
Die Nutzung des entfernten Zugriffs wird nur empfohlen, wenn auf entsprechend geschultes Personal zurückgegriffen werden kann.
Die erforderlichen Kenntnisse werden bei den Betreibern der Anwendung vorausgesetzt. Support kann nur eingeschränkt gewährleistet werden.

1.1 Remote-Zugriff aktivieren und konfigurieren

⚠ **Aus Sicherheitsgründen sollten folgende Sicherheitsmaßnahmen bereits umgesetzt worden sein, sobald der entfernte Zugriff über das Netzwerk bzw. durch andere Rechner aktiviert wird:**

- Konfiguration der → **TLS-/SSL-Verschlüsselung** (Serverzertifikat installieren; Identifikation von ETR gegenüber den Clients)
- Aktivierung der → **Client-Authentifizierung** (nur berechtigte und authentifizierte Clients dürfen auf ETR zugreifen)

Sind diese Vorbereitungen abgeschlossen, kann der Remote-Zugriff in der Konfigurationsdatei "**application.yml**" mit Hilfe des Schlüssels "etr.access.allowFrom" (d.h. im Abschnitt "etr" → Unterabschnitt "access" → dort der Schlüssel "allowFrom") aktiviert und konfiguriert werden.

Im Auslieferungszustand nimmt die ETR-Anwendung nur Anfragen vom lokalen Rechner entgegen. TLS/SSL und die Authentifizierung sind standardmäßig deaktiviert.

Der Abschnitt "etr" ist im Auslieferungszustand in der Konfigurationsdatei bereits vorhanden. Unterhalb dieses Abschnitts können der Unterabschnitt "access" und der benötigte Schlüssel "allowFrom" neu eingefügt werden. Die Vorgaben des YAML-1.1-Dateiformats zu Einrückungen und geschützten Sonderzeichen sind dabei zu beachten.

```
etr:
  access:
    allowFrom: '127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1'

    # ... hier folgen mit entsprechender Einrückung weitere Einträge des
Abschnitts "etr" wie "gruppenModus", "sicherheit" usw.
```

Codeblock 1 Beispiel: Angabe "etr.access.allowFrom" in "application.yml"

1.1.1 Zulässige Werte

Als Wert für den Schlüssel "etr.access.allowFrom" wird ein regulärer Ausdruck (in Java-Pattern-Syntax) angegeben, der definiert, welche externen Hosts (entweder IP-Adressen oder IP-Adressbereiche, ggf. mehrere in Kombination) Zugriff auf die Anwendung erhalten sollen, d.h. freigeschaltet werden.

Der angegebene Ausdruck sollte in Abhängigkeit von der konkreten Netzwerkkonfiguration gewählt werden. In Hinblick auf eine größtmögliche Sicherheit sollte er möglichst streng / möglichst konkret sein und nur diejenigen Rechner bzw. IP-Adressbereiche umfassen, die als anfragende Clients in Frage kommen.

Um Probleme mit geschützten Sonderzeichen zu vermeiden wird empfohlen, den regulären Ausdruck wie in den Beispielen angegeben zusätzlich in einfache Anführungsstriche/Apostroph-Zeichen (') einzuschachteln. Diese zusätzlichen Anführungsstriche/Apostroph-Zeichen werden nicht als Bestandteil des Ausdrucks interpretiert.

Beispiel-Werte für "etr.access.allowFrom":

Regulärer Ausdruck	Freigegebene IP-Adressen bzw. IP-Adressbereiche
'127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1'	nur der lokale Rechner (="localhost")
'172\.20\.\d+\.\d+'	alle IPv4-Adressen, die mit dem Präfix "172.20." beginnen
'192\.168\.1\.2|80\.58\.1\.2'	die beiden konkreten IPv4-Adressen "192.168.1.2" und "80.58.1.2"
'172\.20\.0\.[1-3]'	die 3 konkreten IPv4-Adressen "172.20.0.1", "172.20.0.2" und "172.20.0.3"
'2001:0db8:85a3:0000:0000:8a2e:03[60|70]:\d+'	alle IPv6-Adressen, die mit dem Präfix "2001:0db8:85a3:0000:0000:8a2e:0360:" oder "2001:0db8:85a3:0000:0000:8a2e:0370:" beginnen
'.*'	alle externen Hosts sind zugelassen

Als Standardwert, wenn kein anderer Wert angegeben ist, wird der Wert **'127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1'** angenommen. Dieser Standardwert schränkt den Zugriff ausschließlich auf den lokalen Rechner ("localhost") ein.

1.2 TLS-/SSL-Verschlüsselung

Da mit der ETR-Anwendung schützenswerte Daten übermittelt werden, sollte jegliche Kommunikation, die über das Netzwerk von/zur ETR-Anwendung gesendet/empfangen wird, verschlüsselt sein (TLS = "transport layer security" bzw. SSL = "secure sockets layer").

In ETR kann daher ein TLS-/SSL-Server-Zertifikat konfiguriert werden, anhand dessen sich der ETR-Server gegenüber zugreifenden Clients identifiziert. Das zu verwendende Serverzertifikat muss extern generiert und als Keystore im Format "PKCS #12" bereitgestellt werden.

Per Vorgabe ist TLS/SSL deaktiviert, sodass kein Serverzertifikat benötigt wird. Dies ist insbesondere für einfache lokale Installationen mit deaktiviertem → Remote-Zugriff gedacht und ⚠ **sollte nicht verwendet werden, wenn ELSTER-Transfer auf einem Server betrieben wird und von extern erreichbar ist.**

1.2.1 Unterstützte TLS-/SSL-Protokollversionen und Cipher Suites

ETR bietet in der aktuellen Version folgenden Cipher Suites an:

⇨ Protokollversion TLSv1.3

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

- Protokollversion TLSv1.2 (weniger sicher; nur noch aus Kompatibilitätsgründen für ältere Clients unterstützt)
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Die unterstützten Protokollversionen und Cipher Suites werden unter Sicherheitsbetrachtungen dem Stand der Technik angepasst und können sich in zukünftigen ETR-Versionen ändern. Bei Clients, die mehrere der gelisteten Cipher Suites unterstützen, wendet ETR bevorzugt die als technisch sicherste bekannte Cipher Suite an.

1.2.2 Erzeugung und Installation eines selbstsignierten Serverzertifikats

Um TLS/SSL einzurichten und zu aktivieren, wird zunächst ein entsprechendes Server-Zertifikat benötigt.

- Ist bereits für den Rechner, auf dem ETR betrieben wird, ein passendes Zertifikat von einer offiziellen "Zertifizierungsstelle" erstellt worden, sollte dieses verwendet werden. Das vorhandene Zertifikat kann, wenn es im "PKCS #12"-Format vorliegt (oder ggf. in dieses Format konvertiert wird), direkt → in ETR installiert und TLS/SSL aktiviert werden; d.h. der folgende Befehl zur Erzeugung eines "selbstsignierten" Zertifikats ist in diesem Fall nicht notwendig und wird übersprungen!
- Alternativ *kann* ein Serverzertifikat mit dem folgenden Befehl selbst generiert werden.
  ⚠ Nachteilig ist bei sogenannten "selbstsignierten" Zertifikaten, dass die meisten Clients (u.a. alle modernen Versionen der Webbrowser) diesen Zertifikaten aus Sicherheitsgründen nicht automatisch "vertrauen" d.h. beim Zugriff entsprechende Sicherheitswarnungen auftreten oder manche Clients bzw. Webbrowser diese Art des Zugriffs grundsätzlich unterbinden.

Der folgende Befehl generiert ein *selbstsigniertes* Zertifikat und gibt dieses in einer Zertifikatsdatei (Keystore im "PKCS #12"-Format mit Erweiterung "`.p12`") aus.

```
keytool -genkeypair -alias [Name] -keyalg RSA -keysize
4096 -dname "CN=[Hostname]" -validity 365 -storetype
PKCS12 -keystore [Name].p12 -storepass [Passwort]
```

Die Begriffe in eckigen Klammern stellen Platzhalter dar, die bei der Eingabe ersetzt werden müssen (die eckigen Klammern bitte entfernen!):

- "[Name]" kann ein beliebiger Name für die Zertifikatsdatei sein.
- "[Hostname]" muss der Hostname des Servers sein, auf dem die ETR-Anwendung läuft.
- "[Passwort]" ist ein selbstgewähltes Passwort für den Keystore.

(Das benötigte "`keytool`" wird in ETR im Unterverzeichnis "jre\bin" bzw. "jre/bin" mitgeliefert. Alternativ kann das entsprechende "`keytool`" einer anderen aktuellen Java-Runtime verwendet werden.)

1.2.3 Installation des vorhandenen Zertifikats und Aktivierung von TLS/SSL

In der Konfigurationsdatei "application.yml" die TLS/SSL-Konfiguration anpassen:

```
server:
  ssl:
    enabled: true
    key-store: "[Dateipfad]"
    key-store-password: "[Passwort]"
```

- `key-store` – muss der Dateipfad zum in Punkt 1 erstellten Keystore sein
- `key-store-password` – Ausschließlich für den Windows-Desktop-Betrieb. Die Datei und somit auch das Passwort kann von jedem Windows-Benutzer gelesen werden.

Für andere Installationen wird das Passwort zum Keystore in einer separaten Datei gespeichert. Weitere Informationen finden Sie im Abschnitt "Angabe von Passwörtern bei der Konfiguration".

3) Falls die Anwendung schon läuft, muss sie nach diesen Änderungen neugestartet werden, damit die Änderungen übernommen werden.

1.3 Client-Authentifizierung

ETR bietet die Möglichkeit, den Zugriff durch Clients durch Authentifizierung und Autorisierung (Berechtigungsprüfung) der eingehenden Anfragen einzuschränken. Bei "Serverbetrieb" mit Zugriffen von außerhalb des lokalen Rechners sollte deshalb aus Sicherheitsgründen die Funktion aktiviert werden (=eine Authentifizierungsart ungleich "keine" wählen), um sicherzustellen, dass keine unberechtigten oder möglicherweise böswilligen Clients Daten aus ETR abfragen bzw. an ETR senden. "Zugriffe" können entweder manuelle Anfragen von einem physischen Nutzer (z.B. die Bedienung der Weboberfläche) oder automatisierte Zugriffe eines anderen Rechners (z.B. per REST-API) sein.

1.3.1 Authentifizierungsarten (Überblick)

Konfigurierte Authentifizierungsart (in "application.yml")	**"keine"**	**"yaml"**
Benutzerlogin erforderlich?	nein (anonymer Zugriff)	ja
Art der Authentifizierung	–	"Basic Auth" (nach RFC 7617)
Unterscheidung nach Berechtigungen?	nein (*jeder* Benutzer hat *alle* Berechtigungen)	ja (per externer YAML-Datei konfigurierbar)
"Gruppen-Modus" unterstützt?	nein	ja (per externer YAML-Datei konfigurierbar)

Die bis einschließlich ETR 3.6.0 verwendete Authentifizierungsart mit Hilfe einer externen ".properties"-Datei ist ab ETR-Version 24.04 nicht mehr möglich. Bestehende Installationen sollten auf die Authentifizierungsart "yaml" umgestellt werden.

Zukünftig kann die Unterstützung weiterer Authentifizierungsarten folgen.

1.3.2 Authentifizierungsart "keine"

Im Auslieferungszustand findet in ETR *keine Authentifizierung* der Zugriffe durch Clients statt (*jeder* Benutzer hat *alle* Berechtigungen).

Das eingestellte Authentifizierungsverfahren unterhalb des Schlüssels "*etr.sicherheit.authentifizierung*" in der Konfigurationsdatei "application.yml" ist daher standardmäßig auf "keine" gesetzt. Alle weiteren Einstellungen zur Authentifizierung/Autorisierung werden ignoriert.

```
etr:
  sicherheit:
    authentifizierung: keine
```

Dieser Modus ist insbesondere für einfache lokale Installationen gedacht und **sollte nicht verwendet werden, wenn ELSTER-Transfer auf einem Server betrieben wird und von extern erreichbar ist.**

1.3.3 Authentifizierungsart "yaml"

Bei dieser Authentifizierungsart erfolgt die Hinterlegung der Benutzer, die auf ETR zugreifen können, zusammen mit weiteren Einstellungen in Form einer externen Datei im YAML-1.1-Format. Diese Datei soll von einer privilegierten Person fortlaufend gepflegt werden (z.B. mit einem Texteditor).

Es handelt sich *nicht* um die Konfigurationsdatei "application.yml", sondern um eine zusätzlich zu erstellende YAML-Datei mit beliebigem Namen (empfohlene Dateiendung ".yml" oder ".yaml"). Beide Dateien verwenden das gleiche Dateiformat (YAML), Dateiinhalt und Struktur der Schlüssel unterscheiden sich aber.

1) Die YAML-Datei muss mit passender Struktur im Dateisystem angelegt und mit den benötigten Daten befüllt werden. Mindestens 1 Nutzer mit technischer Rolle "ADMIN" sollte konfiguriert werden, um die grundlegenden "Einstellungen" zu verwalten.

- Login-Namen, den die Clients beim Aufruf von ETR übergeben, um sich zu authentifizieren.
  (z.B. "`beispielnutzer`")
- Hashwerte der Passwörter
  (z.B. Hashwert "`$2y$12$Lj941LUEav3mflaWvwAeCexCNEhcWNg1SblwfpXv1gem1457CwPi2`" zum Passwort "abc", welches der Benutzer z.B. beim Aufruf von ETR im Webbrowser angibt)
  - Passwörter können bis zu 72 Zeichen lang sein. Aus Sicherheitsgründen wird empfohlen, komplexe Passwörter mit mindestens 10 Zeichen Länge zu wählen.
  - Zur Erzeugung der Passwort-Hashes muss das Hashverfahren "bcrypt" verwendet werden. Die erzeugten Hashes werden dann in die Datei eingetragen, sodass die Passwörter nicht im Klartext lesbar sind.
  - Dieses sichere Verfahren generiert bei mehrmaliger Ausführung für ein und dasselbe Passwort als Ausgangswert i.d.R. verschiedene Hashwerte. Dieses Verhalten ist im Verfahren begründet, dient zur Erhöhung der Sicherheit und stellt keinen Fehler dar.
- Berechtigungen (technische Rollen), möglich sind:
  - USER (=Vorgabewert)
    - alle Berechtigungen der beiden folgenden Rollen "USER_STEUERVERWALTUNG" und "USER_POSTFACH"
  - USER_STEUERVERWALTUNG (=Teilmenge der Funktionen von "USER")
    - Zugriff auf die Funktionen zum "Datenaustausch mit der Steuerverwaltung" (Sende- und Abholaufträge)
      - per Weboberfläche
      - per REST-API
      - je nach konfiguriertem "Gruppen-Modus" jeweils Zugriff entweder auf alle Datensätze oder nur die der eigenen Gruppe (+ Standardgruppe)
    - *kein* Zugriff auf die Funktionen für das "Zentrale Unternehmenspostfach" (Bereitstellung von Dokumenten, Rückantworten im "Posteingang")
  - USER_POSTFACH (=Teilmenge der Funktionen von "USER")
    - *kein* Zugriff auf die Funktionen zum "Datenaustausch mit der Steuerverwaltung" (Sende- und Abholaufträge)
    - Zugriff auf die Funktionen für das "Zentrale Unternehmenspostfach" (Bereitstellung von Dokumenten, Rückantworten im "Posteingang")
      - per Weboberfläche
      - per REST-API

        - je nach konfiguriertem "Gruppen-Modus" jeweils Zugriff entweder auf alle Datensätze oder nur die der eigenen Gruppe (+ Standardgruppe)
    - ~~USER_POSTFACH_2.0~~ *(entfällt zukünftig)*
        - die Rolle USER_POSTFACH_2.0 wurde zu USER_POSTFACH umbenannt, beide Rollen sind gleichwertig
    - ADMIN
        - alle Funktionen der anderen technischen Rollen (USER, ACTUATOR, CLI)
        - Zugriff auf privilegierte Funktionen wie die System-"Einstellungen" (u.a. ELSTER-Zertifikatsauswahl) und den "Konfigurationstest"
        - Zugriff auf alle Datensätze, unabhängig vom "Gruppen-Modus"
        - Zugriff auf H2-Datenbank-Web-UI (falls per Konfiguration aktiviert)
    - ACTUATOR
        - Zugriff auf alle technischen Statusinformationen der Anwendung (URLs unterhalb "/actuator/health")
        - aus Sicherheitsgründen *kein* Zugriff auf andere Funktionen (Weboberfläche/REST-API)
    - ~~CLI~~ = e*hemals zur Verwendung mit dem ETR-Kommandozeilen-Client ("ETR-Konsole", "CLI"), wird zukünftig entfallen und sollte nicht mehr verwendet werden*

⇨ Name bzw. Token der → "Gruppe", welcher der Nutzer zugeordnet wird. Wird der "Gruppen-Modus" aktiviert, dürfen Nutzer nur die Daten der gleichen Gruppe einsehen.

Die Versionsangabe im Kopf der Datei ("fileFormat") ist für spätere Erweiterungen vorgesehen und in der aktuellen ETR-Version immer fest 1.

Die Rolle "USER" sollte nach Möglichkeit auf die strengere Rolle "USER_STEUERVERWALTUNG" oder "USER_POSTFACH" geändert/eingeschränkt werden, sofern ein Benutzer nicht beide Funktionalitäten "Datenaustausch mit der Steuerverwaltung" und "Zentrales Unternehmenspostfach" benötigt.

Nach den Vorgaben des YAML-Dateiformats gibt es grundsätzlich verschiedene gleichwertige Möglichkeiten, die enthaltenen Informationen zu kodieren/zu formatieren, um sie mehr oder weniger platzsparend zu beschreiben. Zwecks Übersichtlichkeit empfiehlt sich die "inline"-Darstellung, d.h. Angaben zu einem Benutzer in jeweils 1 Zeile.

```
fileFormat: 1
users:
  # Beispiel 1: ADMIN-Benutzer mit Vollzugriff auf alle Daten und
Funktionen in ETR
  - { login: "Super_User", rolle: "ADMIN", credentials: { passwortHash:
"$2a$10$qX/3feUx04rVRMjjBMDOZuD//vY9YHsmMs3Fn6yLuZwYkiqBf8wH2" }, gruppe:
"Administratoren" }
  # Beispiel 2: normaler Benutzer (Rolle "USER" ist explizit angegeben,
als "gruppe" wird ein extern generiertes Token verwendet)
  - { login: "Hans.Mustermann", rolle: "USER", credentials: {
passwortHash:
"$2a$10$Tf2GGkPbYfZBWJZbAsMtDenMt2w6t2GQs8h5TptGZ0wcnsfUi7W26" }, gruppe:
"{d0b401b8-2ef1-4a94-936a-d501b34c6569}" }
  # Beispiel 3: normaler Benutzer (Rolle ist implizit "USER", als "gruppe"
wird ein sprechender Name verwendet)
  - { login: "Mira.Bellenbaum@musterstadt.beispiel.de", credentials: {
passwortHash:
"$2a$10$LjWRzJQyHQz3v2b0FDFORe44TjNGoac3gWI5MNIQ/Rg4yq4XKVnDi" }, gruppe:
"12/ABC Gemeinde Musterstadt" }
  # Beispiel 4a: Benutzer, nur mit Berechtigung zum "Datenaustausch mit
der Steuerverwaltung" (und gleiche "gruppe" wie in Beispiel 3 - beide
Nutzer können gegenseitig die von ihnen erfassten Sendeaufträge einsehen)
  - { login: "Lieschen.Mueller@musterstadt.beispiel.de", rolle:
"USER_STEUERVERWALTUNG", credentials: { passwortHash:
"$2a$10$6nM5mCyeTYRMmFF63jvHh.3Zxt7rrrutjgTq7.I4k/LwY0jjswziK" }, gruppe:
"12/ABC Gemeinde Musterstadt" }
  # Beispiel 4b: Benutzer, nur mit Berechtigung für das "Zentrale
Unternehmenspostfach" (gleiche "gruppe" wie in Beispiel 3 - beide Nutzer
können gegenseitig die von ihnen erfassten Bereitstellungsaufträge und
zugehörige Rückantworten einsehen)
  - { login: "Lisa.Bonn@musterstadt.beispiel.de", rolle: "USER_POSTFACH",
credentials: { passwortHash:
"$2a$10$Nxh6.jerBD50SD7WryfeP.aQqaWqIn8AjM8WsXYcvQDnbSNLDr2Qi" }, gruppe:
"12/ABC Gemeinde Musterstadt" }
  # Beispiel 5: normaler Benutzer (als "gruppe" wird ein sprechender Name
verwendet, aber anders als in den Beispielen 3/4 - keine gegenseitige
Einsicht der Daten möglich)
  - { login: "EMuAnGe29", credentials: { passwortHash:
"$2a$10$.tp1.5hKgHiVDg3lPv0LNuLf4d/7p3DQQoaTmWw5l342d4K0Um3su" }, gruppe:
"08/XYZ Andere Gemeinde" }
  # Beispiel 6: Technischer Benutzer mit "ACTUATOR"-Rolle ausschließlich
zur automatisierten Systemüberwachung (z.B.
https://<server>:<port>/actuator/health)
  - { login: "monitoring", rolle: "ACTUATOR", credentials: { passwortHash:
"$2a$12$Vb2XFcGcWW7ekoFyXV1.X.gv3Va4K7KMZQxA..If3p0P/aOtmCFHO" }, gruppe:
"Monitoring" }
```

Codeblock 2 Beispieldatei "EXAMPLE-users.yml" mit ETR-Benutzerinformationen

Das kommentierte Beispiel veranschaulicht die Struktur der Datei. Eine entsprechende Datei namens "EXAMPLE-users.yml" wird außerdem in der Installation mitgeliefert (bei Docker-Installationen per `tar`-Befehl ggf. aus dem Container auspacken). Sie kann als Vorlage für die produktive Installation dienen, wenn die Beispieldaten durch die wirklichen Daten ersetzt und vervollständigt werden.

Hinweise zum YAML-Dateiformat
Je nach Anwendungsszenario kann aufgrund praktischer Gesichtspunkte auch eine andere zulässige Kodierung/Formatierung des YAML-Formats als das hier gezeigte "inline"-Beispiel erfolgen. ETR setzt hierzu lediglich voraus, dass die eingelesene Datei eine gültige YAML-1.1-Syntax aufweist. Die für das YAML-Format standardisierten Vorgaben (u.a. Einrückungen, Klammerung, zu schützende Sonderzeichen) sind in jedem Fall zu beachten.
Viele externe Werkzeuge, Datenbanken und Texteditoren bieten eine sehr gute Unterstützung für das manuelle Bearbeiten von YAML-Dateien (u.a. Syntax-Highlighting, automatische Validierung). Einige externe Datenquellen bieten auch die Möglichkeit an, YAML oder mit YAML kompatible Datenformate (z.B. bestimmte JSON-Dialekte) über Konvertierungs- und Exportfunktionen zu erzeugen.

2) Die erzeugte YAML-Datei mit den hinterlegten Benutzerinformationen sollte in einem Verzeichnis angelegt werden, das für den Betriebssystem-Nutzer lesbar ist, mit dem ETR gestartet werden soll.

- Es wird empfohlen, die Berechtigungen auf diese Datei auf Betriebssystemebene möglichst restriktiv zu setzen, sodass der Zugriff möglichst nur durch die Anwendung (ETR) sowie durch Personen, die mit der Pflege der Datei betraut sind, möglich ist.
- Unter Docker kann die Datei (je nach Speicherort) beim Anlegen des Containers ggf. aus einer externen Quelle in den Container gemappt werden.

3) In der "application.yml"-Konfigurationsdatei wird der Ablageort der erzeugten YAML-Datei unter "*etr.sicherheit.yaml.credentials-file-path*" angegeben und zusätzlich die YAML-Authentifizierung per "*etr.sicherheit.authentifizierung*" mit Wert "yaml" aktiviert:

```
etr:
  sicherheit:
    authentifizierung: yaml
    yaml:
      credentials-file-path: "C:\\Beispiel-Konfiguration\\ETR-
Nutzer.yaml"
```

4) Um ein definiertes Verhalten von ETR zu gewährleisten, muss die Anwendung, falls sie bereits läuft, nach Änderungen an der "application.yml" oder in der YAML-Datei mit den hinterlegten Benutzerinformationen neugestartet werden.

1.4 Einschränkung der Zugriffe anhand des Netzwerk-Interfaces (der IP-Adresse)

Für den Fall, dass ETR auf einem Server betrieben wird, der über mehrere Netzwerk-Interfaces (bzw. IP-Adressen) zugreifbar ist, kann die Angabe eines konkreten Netzwerk-Interfaces (bzw. einer konkreten IP-Adresse) erfolgen, sodass der Zugriff *ausschließlich* über diesen Kommunikationsweg erfolgt. Über die anderen Netzwerk-Interfaces (bzw. IP-Adressen) ist ETR dann nicht zugreifbar. Dazu muss in der Konfigurationsdatei "application.yml" der Schlüssel "server.address" mit einer passenden IP-Adresse eingetragen werden.

Der Abschnitt "server" ist im Auslieferungszustand in der Konfigurationsdatei "application.yml" bereits vorhanden (u.a. kann über den Schlüssel "server.port" der Port geändert werden). Der benötigte Schlüssel "address" muss im vorhandenen Abschnitt neu eingefügt werden. Die Vorgaben des YAML-1.1-Dateiformats zu Einrückungen und geschützten Sonderzeichen sind dabei zu beachten.

Beispiel mit IP-Adresse "192.168.50.9":

```
server:
  address: '192.168.50.9'

  # ... hier folgen mit entsprechender Einrückung weitere Einträge des
Abschnitts "server" wie "port", "ssl" usw.
```

1.4.1 Zulässige Werte

Es muss eine gültige IPv4- oder IPv6-Adresse eingegeben werden, die für eines der auf dem Server installierten Netzwerk-Interfaces (z.B. statisch oder per DHCP) vergeben ist. Die Angabe von Multicast-Adressen ist nicht zulässig.

Um Probleme mit geschützten Sonderzeichen zu vermeiden wird empfohlen, die IP-Adresse wie in den Beispielen angegeben zusätzlich in einfache Anführungsstriche/Apostroph-Zeichen (') einzuschachteln. Diese zusätzlichen Anführungsstriche/Apostroph-Zeichen werden nicht als Bestandteil der IP-Adresse interpretiert.

Beispiel-Werte für "server.adress":

IP-Adresse	Beschreibung
`'0.0.0.0'` *oder* `'::'`	ELSTER-Transfer lauscht auf allen lokalen Adressen, sowohl IPv4 als auch IPv6 (welche der beiden Schreibweisen gewählt wird, ist unerheblich)
'127.0.0.1'	Beschränkung des Zugriffs über das sogenannte "Loopback-Interface" auf "localhost" d.h. nur das System, auf dem ETR installiert ist, selbst. Clients auf anderen Systemen (auch nicht im lokalen Netzwerk) können auf ETR nicht zugreifen.
'::1'	IPv6-Variante für "localhost" (entspricht "127.0.0.1" in IPv4)
'192.168.50.9'	ELSTER-Transfer ist ausschließlich über das Netzwerk-Interface mit der IPv4-Adresse "192.168.50.9" erreichbar. (In diesem Beispiel wird angenommen, dass einem der Netzwerk-Interfaces auf dem Server, auf dem ETR installiert ist, zuvor diese Adresse "192.168.50.9" zugewiesen wurde.)
'2001:0db8:85a3:0000:0000:8a2e:0360'	ELSTER-Transfer ist ausschließlich über das Netzwerk-Interface mit der IPv6-Adresse "2001:0db8:85a3:0000:0000:8a2e:0360" erreichbar (analog zum vorherigen Beispiel, hier mit IPv6-Adresse).

Als Standardwert, wenn kein anderer Wert angegeben ist, gilt die Bindung an "alle lokalen Adressen" (analog zum Wert '**0.0.0.0**'). Damit könnte von allen anderen Systemen, die einen Netzwerkzugriff auf den Server haben, auf die ETR-Anwendung zugegriffen werden. Der Remote-Zugriff ist allerdings im Default durch eine Filterung der Remote-IP-Adressen unterbunden.

1.4.2 Alternativen

Alternativ zur Beschränkung des Zugriffs durch Konfiguration per "server.address" direkt in der Anwendung kann die Zugriffsbeschränkung auch auf Infrastrukturebene im Netzwerk (z.B. durch Konfiguration vorhandener Router, Firewalls, Reverse Proxies etc.) erfolgen.

1.5 Betrieb hinter einem Reverse Proxy

1.5.1 Kontextpfad

ELSTER-Transfer unterstützt die Möglichkeit, den Kontextpfad, in welchem die Webanwendung läuft, zu verändern. Standardmäßig ist dieser leer, so dass die Startseite von ELSTER-Transfer unter `/start` erreichbar ist. Wenn ELSTER-Transfer über eine Domain erreichbar sein soll, über die mehrere Webanwendungen laufen, kann ein Kontextpfad in der `application.yml` angegeben werden. Wird der Kontextpfad beispielsweise mit `/etr` definiert, so kann die Startseite über `/etr/start` erreicht werden.

Anpassung in der `application.yml`:

```
server:
  servlet:
    context-path: /etr
```

Der Kontextpfad beginnt mit einem Schrägstrich "/" und darf nicht mit einem Schrägstrich enden. Bitte beachten Sie, dass der Eintrag im Windows-Startmenü "ETR-Browser" mit dieser Einstellung nicht mehr funktioniert und angepasst werden muss.

1.5.2 Auswerten der RFC 7239 Forwarded Headers

Damit ELSTER-Transfer stets zuverlässig den Hostnamen, unter welchem die Anwendung aufgerufen wird, ermitteln kann, müssen bei einem Betrieb hinter einem Reverse Proxy die Forwarded Headers ausgelesen werden.

Dies wird erreicht mit einer Anpassung in der `application.yml`:

```
server:
  forward-headers-strategy: NATIVE
```

Weitere Informationen unter: https://docs.spring.io/spring-boot/docs/3.1.6/reference/html/howto.html#howto.webserver.use-behind-a-proxy-server.tomcat

1.6 Gruppen-Modus

Ab ETR 24.04 besteht die Möglichkeit, von bestimmten Gruppen von Benutzern bzw. Clients erfasste Daten getrennt zu verarbeiten. Die Trennung der Daten erfolgt für per Weboberfläche oder REST-API neu angelegte Sende- und Bereitstellungsaufträge bei aktivierter → **Client-Authentifizierung** (Art der Authentifizierung ungleich "keine") automatisch, sofern in der Liste der zulässigen Benutzer entsprechende Gruppeninformationen hinterlegt sind d.h. jedem Benutzer/Client eine konkrete Gruppe zugewiesen wird. Auch bei den automatisch im "Posteingang" abgeholten Rückantworten zu einem zuvor bereitgestellten Dokument (z.B. einem Bescheid) versucht ETR, die Antworten dem ursprünglichen Dokument zuzuordnen und die dort hinterlegten Gruppenzuordnung anzuwenden.

Nach welchen Kriterien die Gruppen gebildet werden und welche Information (Gruppenname, organisatorische Bezeichner, Token usw.) zur Identifikation der Gruppen verwendet wird, ist grundsätzlich vom jeweiligen Anwendungskontext abhängig und daher im Rahmen der technischen Mindestanforderungen (Länge der Bezeichner min. 3 bis max. 255 Zeichen) frei wählbar. Ein häufiger Anwendungsfall ist die Trennung nach organisatorischen Merkmalen (z.B. Behörden, Stellen, Gebietskörperschaften).

Abgesehen von den beschriebenen → Ausnahmen gilt dabei die Regel:

- **Die Vergabe gleicher Gruppennamen bzw. Gruppen-"Token" zu einem Benutzer/Client ordnet in ETR die zugehörigen Daten der gleichen Gruppe zu.**

1.6.1 Aktivieren/Deaktivieren des "Gruppen-Modus"

Ob bei Zugriff auf ETR per Weboberfläche oder per REST-API die Daten nach Gruppen unterschieden werden sollen, kann global per Konfiguration in der Datei `application.yml` (Schlüssel "etr.gruppenModus") eingestellt werden.

```
etr:
  gruppenModus: gruppen_token
```

Es gibt derzeit 2 Wahlmöglichkeiten.

Einstellung mit Wert "keine_gruppen" (dies ist die Vorgabe):

- "Gruppen-Modus" deaktiviert. Alle Benutzer können auf alle Daten zugreifen, unabhängig von der Gruppenzuordnung der Daten und den Rollen/Berechtigungen der Nutzer/Clients. Eine eventuelle trotzdem vorgenommene Gruppenzuordnung wird ignoriert.
- Dies entspricht dem Verhalten in älteren ETR-Versionen.

Einstellung mit Wert "gruppen_token":

- "Gruppen-Modus" aktiviert. Bei jedem Zugriff per Weboberfläche oder REST-API wird geprüft und sichergestellt, dass jeder Benutzer/Client nur auf die Daten der ihm zugeordneten Gruppe zugreifen kann. Datensätze, die anderen Gruppen zugeordet sind, sind für den jeweiligen Benutzer/Client nicht sichtbar. Dabei gibt es aber → Ausnahmen.
- Dazu muss gleichzeitig die → **Client-Authentifizierung** konfiguriert und aktiviert sein. Dies wird beim Start von ETR geprüft und der Start ggf. abgebrochen.
- Allen Benutzern/Clients müssen Gruppen zugeordnet sein (bei Verwendung der YAML-Datei im Feld "gruppe" angegeben). Die angegebenen Gruppen sollten korrekt sein.

(In zukünftigen ETR-Versionen können ggf. weitere Arten der Unterscheidung von Gruppen eingeführt werden.)

Wird der zuvor aktivierte Gruppen-Modus per Konfiguration durch Setzen des Werts "keine_gruppen" wieder deaktiviert, bleiben die vorhandenen Gruppen-Zuordnungen für die bereits erfassten Datensätze in ETR erhalten. Lediglich bei der Abfrage der Daten wird das jeweilige "Gruppen-Token" nicht mehr berücksichtigt, d.h. allen Benutzern/Clients werden immer alle Datensätze ohne Unterscheidung nach "Gruppen-Token" angezeigt bzw. zurückgegeben. Auf diese Weise ist es möglich, einen zunächst *temporär* deaktivierten "Gruppen-Modus" später wieder per Konfiguration zu reaktivieren, ohne dass die in ETR bereits gespeicherten Gruppenzuordnungen verloren gehen.

1.6.2 Ausnahmen bei der Gruppenzuordnung

Zuordnung zur "Standardgruppe"

Die Zuordnung von erstellten Sende- oder Bereitstellungsaufträgen (sowie abgeholten Rückantworten zu diesen Aufträgen) zu einer konkreten Gruppe kann in einigen Fällen nicht automatisch erfolgen:

1. bei älteren Sende- oder Bereitstellungsaufträgen, die vor ETR-Version 24.04 erstellt wurden, oder zugehörigen Rückantworten
2. falls Aufträge von Benutzern ohne Angabe einer Gruppe (in der YAML-Datei mit der Liste der zulässigen Benutzer im Feld "gruppe") angelegt werden
   - Dies ist nur möglich, wenn während des Anlegens eines Sende- oder Bereitstellungsauftrags der Gruppen-Modus deaktiviert ist (d.h. die Einstellung "keine_gruppen" verwendet wird).
   - Bei aktiviertem Gruppen-Modus (Wert "gruppen_token") wird von ETR technisch immer eine hinterlegte Gruppenzuordnung erzwungen. Für neue Aufträge ist dieser Fall der Zuordnung zur Standardgruppe damit ausgeschlossen.
3. bei Sende- oder Bereitstellungsaufträgen, die über die Dateiimport-Schnittstelle angelegt werden

Um eine geordnete Weiterverarbeitung zu ermöglichen, werden die genannten Datensätze einer "Standardgruppe" zugeordnet, auf die *alle Benutzer* Zugriff haben.

Benutzer mit ADMIN-Rolle

Benutzer mit ADMIN-Rolle erhalten implizit die Berechtigung, *alle* Daten einzusehen, ohne Berücksichtigung ihrer Gruppenzuordnung. Dies kann z.B. zur Problemanalyse nützlich sein.

Neu angelegte Sende- oder Bereitstellungsaufträge (sowie abgeholten Rückantworten zu diesen Aufträgen) werden dagegen auch für Benutzer mit ADMIN-Rolle, wie für alle Benutzer üblich, der in der YAML-Datei eingetragenen Gruppe zugeordnet. Andere Benutzer ohne ADMIN-Rolle können daher die von einem Benutzer mit ADMIN-Rolle erzeugten Aufträge einsehen, sofern ihnen explizit die gleiche Gruppe zugewiesen wird. Andernfalls sind diese Daten nur für andere Benutzer mit ADMIN-Rolle einsehbar.

Grundsätzlich wird empfohlen, für die Benutzer mit ADMIN-Rolle eine dedizierte Gruppe zu verwenden z.B. eine Gruppe mit dem Namen "Administratoren" oder einem festen Token. Aufgrund der mit der ADMIN-Rolle verbunden Privilegien sollten diese Benutzer darüber hinaus nicht für die normale Arbeit, sondern nach Möglichkeit ausschließlich für technische und organisatorische Arbeiten in ETR verwendet werden.

Nicht-Gruppenbezogene Funktionen

Die Einstellung des "Gruppen-Modus" hat keine Auswirkung auf übergreifende Funktionalität, die nicht im Kontext eines konkreten Benutzers von ETR im Hintergrund ausgeführt wird (z.B. die in ETR integrierte Archivierungsfunktion).

1.6.3 Andere Arten der Zuordnung von Datensätzen

Im Kontext der "Datenabholung von der Steuerverwaltung" ist es technisch nicht ohne Weiteres möglich, die abgeholten Dokumente (Downloads) konkreten Benutzer-"Gruppen" zuzuordnen. Hintergrund ist, dass bei diesen Dokumenten vom Absender in der Regel keine weiteren Informationen zur Adressierung mitgeliefert werden, die zur automatisierten Zuordnung zu Benutzergruppen verwendet werden können, und es sich auch nicht um Rückantworten mit Bezug zu einem in ETR vorhandenen Dokument handelt.

Abhängig von der konkreten Datenart und des Absenders der abgeholten Dokumente kann in vielen Fällen versucht werden, eine Unterscheidung anhand fachlicher Metadaten (z.B. AGS = Amtlicher Gemeindeschlüssel) oder anhand des Dateinamens der abgeholten Nutzdaten im Eingangsverzeichnis vorzunehmen.

1.7 Überblick über die Sicherheitskonfiguration in ETR (Zusammenfassung)

Zu den Themen "Remote-Zugriff, TLS-/SSL-Verschlüsselung und Authentifizierung" können zusammenfassend folgende Einstellungen in der Datei **INSTALL_DIR/config/application.yml** vorgenommen werden:

Konfigurationsschlüssel (in Kurzschreibweise)	**Standardwert**	**Beschreibung**
etr.access.allowFrom	**'127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1'**	Regulärer Ausdruck (in Java-Pattern-Syntax), der definiert, welche externen Hosts (entweder IP-Adressen oder IP-Adressbereiche, ggf. mehrere in Kombination) Zugriff auf die Anwendung erhalten sollen d.h. freigeschaltet werden. Der Vorgabewert schränkt den Zugriff auf "localhost" ein.

Konfigurationsschlüssel (in Kurzschreibweise)	Standardwert	Beschreibung
server.address	**'0.0.0.0'**	Einschränkung der Zugriffe anhand des Netzwerk-Interfaces (der IP-Adresse) für den Fall, dass ETR auf einem Server betrieben wird, der über mehrere Netzwerk-Interfaces zugreifbar ist. Gültige IPv4- oder IPv6-Adresse; der Vorgabewert steht für die Bindung an *alle* lokal vorhandenen Netzwerk-Interfaces.
server.ssl.enabled	**false**	`false` - Zugriff auf Weboberfläche zulässig per HTTP (unverschlüsselt) `true` - Zugriff auf Weboberfläche zulässig per HTTPS (verschlüsselt) Bei Aktivierung von TLS/SSL ist zu beachten, dass die weiteren TLS/SSL-Optionen (Installation eines Zertifikats) entsprechend konfiguriert werden müssen. Aktuell in ETR verwendetes Sicherheitsprotokoll ist TLS 1.3 mit den unterstützten Cipher-Suiten TLS_AES_128_GCM_SHA256 und TLS_AES_256_GCM_SHA384 (siehe → Unterstützte TLS-/SSL-Protokollversionen und Cipher Suites)
server.ssl.key-store	*(leer)*	Pfad zu einer PKCS#12-Schlüsseldatei mit öffentlichem und privatem Schlüssel (z.B. mein_zertifikat.p12) im Dateisystem, deren Zertifikat als Server-Zertifikat bei aktivierter TLS-/SSL-Option verwendet wird. Standardmäßig leer, da `server.ssl.enabled` standardmäßig deaktiviert ist.
server.ssl.key-store-password	*(leer)*	Zugehöriges Passwort beim Zugriff auf die PKCS#12-Schlüsseldatei, die das Server-Zertifikat enthält. Da das Passwort im Klartext eingetragen wird, sollte der Zugriff auf die Datei `application.yml` im Dateisystem aus Sicherheitsgründen entsprechend beschränkt bleiben.

Konfigurationsschlüssel (in Kurzschreibweise)	Standardwert	Beschreibung
		Standardmäßig leer, da `server.ssl.enabled` standardmäßig deaktiviert ist.
server.servlet.context-path	/	Kontextpfad (als Infix/Bestandteil der URL), unter dem die ETR-Webanwendung verfügbar gemacht wird. Standardmäßig ist dieser leer. Er kann aber z.B. im Serverbetrieb für die Verwendung hinter Reverse-Proxies an die jeweiligen Erfordernisse angepasst werden.
etr.sicherheit.authentifizierung	**keine**	Steuert die Art der Authentifizierung bei Zugriff über die Weboberfläche oder REST-Schnittstelle. Zulässige Werte: ⇨ "keine" (Vorgabewert) • Keine Authentifizierung notwendig. Alle Zugriffe erfolgen anonym. • Es findet keine Berechtigungs-/Rollenprüfung statt (*jeder* Benutzer darf *alle* ETR-Funktionen nutzen) ⇨ "yaml" • Für alle Zugriffe auf ETR z.B. per Web-Oberfläche wird eine Authentifizierung per "Basic Auth" (nach RFC 7617: Benutzername/Kennwort) verlangt. • Erlaubte Login-Namen, Hashwerte der Passwörter, Berechtigungen, zugeordnete "Gruppen" etc. sind in einer externen Datei im YAML-Format hinterlegt. • Die Angabe des externen Dateipfads ist unter "etr.sicherheit.yaml.credentials-file-path" erforderlich.
etr.sicherheit.yaml.credentials-file-path	*(leer)*	Pfad zur externen YAML-Konfigurationsdatei mit den erlaubten Login-Namen,

Konfigurationsschlüssel (in Kurzschreibweise)	Standardwert	Beschreibung
		Hashwerte der Passwörter, Berechtigungen, zugeordnetes "Gruppen-Token" etc. im Dateisystem. Notwendig, wenn "etr.sicherheit.authentifizierung" auf den Wert "yaml" gesetzt wurde.
etr.gruppenModus	**keine_gruppen**	Steuert, nach welchem Modus die Nutzer/Clients Zugriff auf gruppenspezifische Informationen erhalten. Details siehe → "Gruppen-Modus". Zulässige Werte: ⇨ "keine_gruppen" (Vorgabewert): Gruppen-Modus deaktiviert. Alle Daten sind für alle Nutzer/Clients zugreifbar, unabhängig von der Gruppenzuordnung der Daten und den Rollen/Berechtigungen der Nutzer/Clients. Entspricht dem Verhalten in älteren ETR-Versionen. ⇨ "gruppen_token": Gruppen-Modus aktiviert. Die Einschränkung des Zugriffs erfolgt durch Bildung von Benutzergruppen (Zuordnung von Gruppennamen bzw. "Gruppen-Token"). Dazu muss gleichzeitig die Client-Authentifizierung (Schlüssel "etr.sicherheit.authentifizierung") konfiguriert werden.

2 Nutzereinstellungen (ELSTER-Zertifikat) konfigurieren

Alternativ zum Formular "Einstellungen" (bei aktivierter Authentifizierung nur als Benutzer mit "ADMIN"-Rolle zugänglich) kann die grundlegende Konfiguration der Anwendung 1:1 auch in der Konfigurationsdatei *application.yml* im Abschnitt "etr.einstellungen" erfolgen.

Alle Änderungen an der *application.yml* werden erst nach einem Neustart von ELSTER-Transfer angewendet.

> Das Setzen der Nutzereinstellungen in der Konfigurationsdatei *application.yml* erleichtert das Einrichten von Windows-Server-, Linux- und Docker-Installationen von ELSTER-Transfer, die von mehreren Nutzern verwendet werden. Für die Windows-Desktop-Installation ist eine Vorkonfiguration der Nutzereinstellungen *nicht* vorgesehen.

2.1 Aktivieren der Vorkonfiguration

Sobald in der Konfigurationsdatei unter **INSTALL_DIR/config/application.yml** das Zertifikat hinterlegt ist, wird diese Einstellung aktiv und auch die anderen Nutzereinstellungen können nur noch über die Konfigurationsdatei gesetzt werden. Die Konfiguration über die Weboberfläche ist dann nicht mehr möglich und das Formular "Einstellungen" beschränkt sich ausschließlich auf die Anzeige.

Ausschließlich nur andere Nutzereinstellungen in der "application.yml" zu setzen (ohne das Zertifikat festzulegen), ist nicht möglich.

2.2 Übersicht der Einstellungen

Einstellung	Standardwert	Beschreibung

etr.einstellungen.elsterZertifikat.dateiPfad	*(leer)*	Absoluter Pfad zu der ELSTER-Zertifikatsdatei.
		Um einen relativen Pfad zum Datenverzeichnis anzugeben, kann der Platzhalter "${etr.datenverzeichnis}" verwendet werden. Die Verwendung von Umgebungsvariablen ist ebenso möglich.
		Eingabe für Windows z.B.:
		`"C:\\etr\\elster-zertifikat.pfx"`
		`"${etr.datenverzeichnis}/elster-zertifikat.pfx"`
		`"${USERPROFILE}/elster-zertifikat.pfx"`
		(wobei 'USERPROFILE' eine Windows-spezifische Umgebungsvariable ist)

Einstellung	Standardwert	Beschreibung
		Eingabe für Linux z.B.: `"/opt/ELSTER-Transfer/etr/config/elster-zertifikat.pfx` Das Zertifikats-Passwort wird in einer getrennten Datei angegeben (Windows-Service: config/etr-01.xml; Linux config/secrets.env). Optional; Die Angabe aktiviert die Vorkonfiguration.
etr.einstellungen.eingangsverzeichnis	"${etr.datenverzeichnis}/eingang"	Absoluter Pfad des Eingangsverzeichnisses. Pflichtfeld, wenn Vorkonfiguration aktiviert.
etr.einstellungen.ausgangsverzeichnis	"${etr.datenverzeichnis}/ausgang"	Absoluter Pfad des Ausgangsverzeichnisses. Pflichtfeld, wenn Vorkonfiguration aktiviert.
etr.einstellungen.importverzeichnis	"${etr.datenverzeichnis}/import"	Absoluter Pfad des Importverzeichnisses. Pflichtfeld, wenn Vorkonfiguration aktiviert.
etr.einstellungen.dateienNachUebermittlungLoeschen	false	Option, ob die Originaldateien von Datenübermittlungen und Bereitstellungen nach dem Absenden aus dem Ausgangsverzeichnis gelöscht werden sollen. `false` – Dokumente bleiben erhalten `true` – Dokumente werden gelöscht Pflichtfeld, wenn Vorkonfiguration aktiviert.
etr.einstellungen.intervalDauerauftrag	10m	Intervall für die Durchführung der Daueraufträge. Eingabe z.B. "1d" (Einmal täglich) oder "1h" (einmal stündlich) oder "30m" (alle 30 Minuten) Pflichtfeld, wenn Vorkonfiguration aktiviert.
etr.einstellungen.netz	internet	Netz, dass verwendet wird, um Netzwerkverbindungen herzustellen. `internet` – Datenverbindungen über das reguläre Internet `ndb` – Datenverbindungen über das Netz des Bundes (NdB-VN, ehem. DOI-Netz) Pflichtfeld, wenn Vorkonfiguration aktiviert.

Einstellung	Standardwert	Beschreibung
etr.einstellungen.proxy.host	*(leer)*	Host-Adresse oder IP-Adresse des zu verwendenden Proxyservers. Falls kein Host eingetragen ist, wird kein Proxyserver verwendet. Wird erst aktiv, wenn die Vorkonfiguration aktiviert ist.
etr.einstellungen.proxy.port	*(leer)*	Port des zu verwendenden Proxyservers. 'port' kann nur bzw. muss immer zusammen mit 'host' verwendet werden. Wird erst aktiv, wenn die Vorkonfiguration aktiviert ist.
etr.einstellungen.proxy.nutzername	*(leer)*	Nutzername für die Authentifizierung an dem Proxyserver, falls notwendig. Das Proxy-Passwort wird in einer getrennten Datei angegeben (Windows-Service: config/etr-01.xml; Linux config/secrets.env) Wird erst aktiv, wenn die Vorkonfiguration aktiviert ist.

3 Angabe von Passwörtern bei der Konfiguration

Um sensible Passwörter besser zu schützen, werden diese unabhängig von der `application.yml`-Datei in einer separaten Datei mit besonders eingeschränkten Zugriffsrechten gespeichert.

Übersichtstabelle möglicher Passwörter in ETR:

Umgebungsvariable	Beschreibung
`ETR_EINSTELLUNGEN_ELSTERZERTIFIKAT_PASSWORT`	Passwort für das in der 'etr.einstellungen.elsterZertifikat.dateiPfad'-Einstellung konfigurierte ELSTER-Zertifikat, falls das ELSTER-Zertifikat bereits in der `application.yml` vorkonfiguriert ist. Weitere Information im Abschnitt "Nutzereinstellungen (ELSTER-Zertifikat) konfigurieren"
`ETR_EINSTELLUNGEN_PROXY_PASSWORT`	Passwort für die Authentifizierung am Proxyserver, der in Einstellungen unterhalb 'etr.einstellungen.proxy' per Vorkonfiguration in der `application.yml` konfiguriert wurde. Weitere Information im Abschnitt "Nutzereinstellungen (ELSTER-Zertifikat) konfigurieren"
`SERVER_SSL_KEY_STORE_PASSWORD`	Passwort für den Zugriff auf den Keystore mit dem (SSL-/TLS-)Web-Server-Zertifikat, welcher unter der 'server.ssl.key-store'-Einstellung in der `application.yml` konfiguriert wurde. Weitere Information im Abschnitt "Remote-Zugriff, SSL-Verschlüsselung und Authentifizierung"
`SPRING_DATASOURCE_PASSWORD`	Passwort für den Datenbankbenutzer der ETR-internen Datenbank mit administrativen Berechtigungen. Entspricht der bzw. übersteuert die 'spring.datasource.password'-Einstellung in der `application.yml`. Details zur Datenbankkonfiguration finden sich auch in den Abschnitten "Konfiguration" bzw. "Anbindung externer Datenbanken". bei Verwendung der internen H2-Datenbank (Vorgabe) Ein Standardpasswort ist vorkonfiguriert. Dieses kann *vor*

Umgebungsvariable	Beschreibung
	dem ersten Start von ETR nach Wunsch geändert werden. Eine spätere Anpassung ist ebenfalls möglich, wenn das Passwort in der dann bereits existierenden H2-Datenbank zusätzlich manuell geändert wird. bei Verwendung einer externen Datenbank (z.B. Postgres) Zu dem in der in der `application.yml` unter 'spring.datasource.username' eingetragenen, vorab in der externen Datenbank angelegten und entsprechend berechtigten Datenbanknutzer gehörendes Passwort.

3.1 Windows (nur Server-Installation)

Wenn die Installation als Dienst unter Windows erfolgte, können die Passwörter in der XML-Konfigurationsdatei *INSTALL_DIR/config/etr-01.xml* der jeweiligen Dienst-Instanz angegeben werden.

Hierfür wird das Passwort in die Vorlage eingetragen:

```
  <env name="ETR_EINSTELLUNGEN_ELSTERZERTIFIKAT_PASSWORT" value="mein-
sicheres-passwort"/>
```

3.2 Linux

Zum Speichern von Passwörtern existiert unter Linux die Datei *INSTALL_DIR/config/secrets.env* im Format einer Umgebungsvariablendatei (d.h. Angaben in der Form "VARIABLE=WERT").

Die Datei wird von Systemd eingelesen, wenn der ETR-Service gestartet wird. Der Zugriff auf die Datei ist auf den root-Nutzer beschränkt.

```
ETR_EINSTELLUNGEN_ELSTERZERTIFIKAT_PASSWORT=mein-sicheres-passwort
```

3.3 Docker

Unter Docker existiert die `secrets.env`-Datei analog, muss jedoch im Rahmen der Installation manuell aus dem Archiv extrahiert werden (siehe "Handbuch Docker").

Die Datei wird unter */etc/etr-01/secrets.env* gespeichert und durch den Docker-Daemon bei dem Start des ETR-Containers eingelesen. Der Zugriff ist auf den root-Nutzer beschränkt.

```
ETR_EINSTELLUNGEN_ELSTERZERTIFIKAT_PASSWORT=mein-sicheres-passwort
```

Hinweis zur Reihenfolge der Konfigurationsschritte
Zum Erstellzeitpunkt des Docker-Containers müssen die Passwörter bereits in der Datei *secrets.env* eingetragen sein. Ein neuer Docker-Container sollte daher erst nach Setzen aller Passwörter erstellt werden. Sollte der Docker-Container bereits vorhanden sein (z.B. im Fall einer Update-Installation), **muss** bei Änderungen eines Passworts der vorhandene Container entfernt und neu erstellt werden.

4 Ändern des Benutzerpassworts einer bestehenden H2-Datenbank

Abgrenzung: Externe Datenbanken
Die Beschreibungen in diesem Abschnitt sind für *externe* Datenbanken (z.B. Postgres) *nicht* zutreffend. Dort erfolgt die Verwaltung der Login-Benutzer (inkl. des Rücksetzens der Passwörter) im Unterschied zur integrierten H2-Datenbank durch den jeweiligen Datenbankadministrator.

Standardmäßig verwendet ETR eine dateibasierte H2-Datenbank im Anwendungsdatenverzeichnis zur Speicherung der internen Daten. Diese H2-Datenbank wird beim ersten Start von ETR angelegt und mit einem Standardpasswort versehen, sofern dieses in der `application.yml`-Konfigurationsdatei nicht nach der abgeschlossenen Installation nach Wunsch abgeändert wurde (siehe Abschnitt "Konfiguration").

Die Änderung des Passworts ist auch nachträglich möglich, indem das neue Passwort *gleichzeitig*

- in die jeweilige Konfigurationsdatei eingetragen wird (und dabei das vorherige Passwort ersetzt)
- in der bestehenden H2-Datenbank per SQL-Befehl geändert wird (siehe folgender Abschnitt)

Das bisherige Passwort des Datenbanknutzers muss dabei bekannt sein (entweder das konfigurierte Standardpasswort oder das zuvor nach der Installation nach Wunsch abgeänderte Passwort).

4.1 Manuelle Passwortänderung über H2-Konsole

Voraussetzungen:

- ETR muss installiert und lauffähig sein. Ob die Systemeinrichtung abgeschlossen wurde (ELSTER-Zertifikat entweder per "Einstellungen" oder per Vorkonfiguration in "application.yml" hinterlegt), spielt keine Rolle.
- *falls die Nutzerauthentifizierung aktiviert wurde ("etr.sicherheit.authentifizierung" ist z.B. "yaml"):*
  ein Nutzer mit "ADMIN"-Rolle muss definiert worden sein
- zu Beginn ist das *alte* Passwort noch konfiguriert
- der ETR-Dienst ist zunächst beendet

Vorgehensweise:

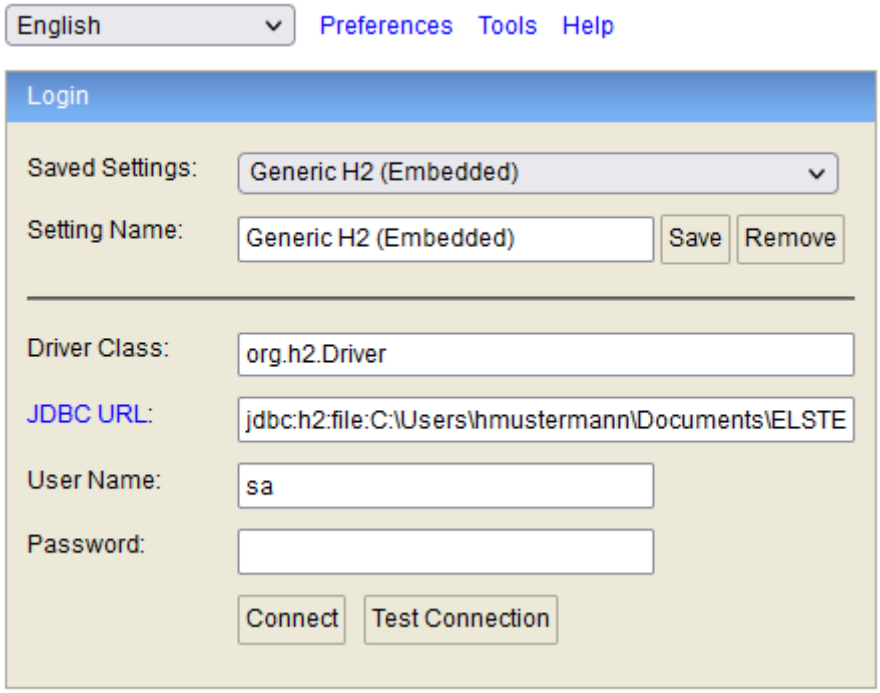
1. In der `application.yml`-Datei die H2-Konsole aktivieren:

```
spring:
  h2:
    console:
      # aktiviert die h2-Konsole
      enabled: true
      # ermöglicht Remote-Zugriff auf die h2-Konsole, wird nur für
    die Docker-Variante von ETR benötigt
      settings:
        web-allow-others: true
```

Codeblock 3 application.yml (Auszug): H2-Konsole aktivieren

2. ETR-Dienst starten → z.B. die Startseite lässt sich öffnen

3. H2-Konsole (http://localhost:8081/h2-console/) öffnen.
   Wenn per "application.yml" die Nutzerauthentifizierung aktiviert ist, muss der Login an der Weboberfläche mit einem Nutzer mit "ADMIN"-Rolle erfolgen.



   - (optional) die Sprache der H2-Konsole kann über die Auswahlliste oben links auf "Deutsch" geändert werden

4. JDBC URL prüfen: die URL sollte den Dateipfad zur h2 Datenbank-Datei beinhalten, jedoch **ohne Dateiendung**.

   - Die URL wird ebenfalls beim Start von ETR bei aktivierter H2-Konsole in die Logdatei geschrieben. Der Logeintrag startet mit "H2 console available at:".
   - Beispiel-URL (Linux):
     `jdbc:h2:file:/home/etr/elster-transfer/daten/elstertransfer`
   - Beispiel-URL (Windows Desktop-Installation):
     `jdbc:h2:file:C:\Users\hmustermann\Documents\ELSTER-Transfer\daten\elstertransfer`
   - Beispiel-URL (Windows Server-Installation):
     `jdbc:h2:file:C:\Users\etr-service-user\Documents\ELSTER-Transfer\etr-01\daten\elstertransfer`

5. Anmeldungsinformationen eingeben und "Test Connection" klicken → "Test successful" erscheint

   - "User Name" ist für ETR standardmäßig "sa" (=Benutzer mit administrativen Berechtigungen u.a. für Ausführung Liquibase-Migration)
   - "Password" ist noch das *alte* Passwort (ggf. das *Standardpasswort*, falls nicht abweichend gesetzt)

6. "Connect" klicken → eine SQL-Eingabeoberfläche erscheint
7. `ALTER USER SET PASSWORD`-SQL-Befehl zur Passwortänderung zusammen mit dem *neuen* Passwort eingeben, danach  "Run" klicken

```
ALTER USER sa SET PASSWORD 'neuesPasswort'
```

Codeblock 4 Beispiel: ALTER USER SET PASSWORD

1. → Bestätigung erscheint (ggf. mit Ausführungszeit, wg. DDL *keine* geänderten Datensätze), *keinesfalls* aber eine Fehlermeldung
8. ETR-Dienst beenden
9. in der `application.yml`-Datei die H2-Konsole wieder deaktivieren (Änderung aus Schritt 1 zurücknehmen)
10. neues Passwort in die Konfiguration eintragen und speichern, je nach Betriebssystem/Installationsart
    a. Windows-Desktop:
    in der `application.yml`-Datei unter "spring.datasource.password" das bestehende alte Passwort ersetzen (falls zuvor explizit ein Passwort konfiguriert war) oder das neue Passwort zusätzlich eintragen *und* zugleich die Option einkommentieren (falls zuvor das Standardpasswort verwendet wurde)
    b. Windows-Server:
    in der Dienst-Konfigurationsdatei (z.B. "etr-01.xml") unter der Umgebungsvariable "SPRING_DATASOURCE_PASSWORD" das bestehende alte Passwort ersetzen
    c. Linux/Docker:
    in der `secrets.env`-Datei unter der Umgebungsvariable "SPRING_DATASOURCE_PASSWORD" das bestehende alte Passwort ersetzen. Nur unter Docker muss zusätzlich der Container entfernt und neu erzeugt werden, damit die Änderungen der `secrets.env`-Datei angewendet werden.
11. ETR-Dienst neu starten → *keine* Fehler im Logfile beim Start
12. Prüfung: Startseite von ETR lässt sich normal öffnen und bedienen

5 Anbindung externer Datenbanken

Standardmäßig verwendet ETR eine dateibasierte H2-Datenbank im Anwendungsdatenverzeichnis zur Speicherung der internen Daten.

Alternativ ist auch die Nutzung folgender externer, auf einem dedizierten Server installierter Datenbanksysteme möglich:

Unterstütztes Datenbanksystem	Mindestversion
H2 (auf dediziertem Server)	>= 2.3.230
Postgres	>= 16

Für diese Datenbanksysteme werden entsprechende Treiber in ETR mitgeliefert. Zukünftig können weitere Datenbanksysteme folgen.

Die Installation und Wartung der externen Datenbank (z.B. Erstellen von Backups, Aktualisierungen des Datenbanksystems, Verwaltung von Zugriffsrechten und Speicherplatz) muss durch eine Datenbankadministrator außerhalb von ETR erfolgen.

Insbesondere muss

- die Datenbank durch ETR netzwerktechnisch erreichbar sein
- eine ausreichende Anzahl an zulässigen eingehenden Datenbankverbindungen konfiguriert sein (in der Standardkonfiguration durch ETR genutzt: 30 gleichzeitige Verbindungen)
- ausreichend Speicherplatz (auch für temporäre Daten in Sitzungen/Transaktionen) für die Datenbank bereitgestellt werden
- ein geeigneter, DIN-91379-konformer Zeichensatz zur Speicherung der Daten verwenden werden (Empfehlung: Zeichensatz "UTF-8", in Postgres als "`UTF8`" benannt, siehe auch RFC 3629)

Bekannte Einschränkungen bei Verwendung externer Datenbanksysteme

- ETR erstellt kein automatisches Backup für externe Datenbanken (eine entsprechende Warnung wird beim Anwendungsstart ausgegeben). Hierfür ist bei externen Datenbanken der Datenbankadministrator zuständig.
- Die Datenübernahme zwischen einer ETR-internen H2-Datenbank und externen Datenbanksystemen wird derzeit *nicht* ausdrücklich unterstützt. Im Rahmen des Handbuchs wird davon ausgegangen, dass externe Datenbanken neu installiert werden.

5.1 Voraussetzungen

Die benötigten (noch leeren) Datenbanken und zugehörige Login-Benutzer mit entsprechenden Berechtigungen müssen vorab vom Datenbankadministrator angelegt werden, damit ETR diese nutzen kann.

Benötigt wird mindestens:

- 1 Login-berechtigter Nutzer
- mit Vollzugriff auf 1 Datenbank-Instanz / 1 Schema, zur ausschließlichen Verwendung durch die jeweilige ETR-Installation

5.2 Konfiguration

Um eine externe Datenbank für ETR zu verwenden, ist es notwendig, die Standardkonfiguration von ETR im Abschnitt "spring.datasource" der `application.yml`-Konfigurationsdatei (und ggf. zugehörige Dateien) anzupassen. Die Kommentarzeichen von im Standard auskommentierten Konfigurationsschlüsseln sind dabei zu entfernen.

- "username" = Name des Datenbanknutzers mit administrativen Berechtigungen auf dem Zielschema (u.a. für Datenbank-Updates/-Migrationen)
- "password" = nur bei *Desktop*-Installationen (s.u.): das zum Datenbanknutzer 'username' gehörende Passwort
- "url" = die konkrete JDBC-Datenbank-URL, beginnend mit Präfix "jdbc:"

Beispiel für eine Desktop-Installation (Auszug einer `application.yml`-Konfiguration):

```
spring:
  datasource:
    # Name des Datenbanknutzers mit administrativen Berechtigungen auf
dem Zielschema (u.a. für Datenbank-Updates/-Migrationen)
    username: etr_nutzer
    # zum Datenbanknutzer 'username' gehörendes Passwort
    password: me1n_geheimes_Passw@rt

    ### Optionen nur relevant für _externe_ Datenbanken (z.B. Postgres)
###
    url: jdbc:postgresql://localhost:5432/etr_daten
```

⚠ Im Unterschied zu Desktop-Installationen wird bei Windows- oder Linux-Server-Installationen das Passwort des Datenbanknutzers aus Sicherheitsgründen als Umgebungsvariable "SPRING_DATASOURCE_PASSWORD" in einer separaten Datei gespeichert, auf die nur ein eingeschränkter Zugriff möglich ist.

- Windows: in der jeweiligen Dienst-Konfigurationsdatei (z.B. "etr-01.xml")
- Linux (inkl. Docker): in der Datei "secrets.env" (neben der `application.yml` im gleichen Ordner)

Weitere Informationen im Abschnitt "Angabe von Passwörtern bei der Konfiguration".

6 Abrufmöglichkeit von "Health"-Zustandsinformationen

Systemvoraussetzung
Diese Funktionalität erfordert ELSTER-Transfer Version 2.2.0 (oder höher).

Der aktuelle Zustand der ETR-Anwendung kann per HTTP-GET-Operation abgefragt werden.

- Aus Sicherheitsgründen werden standardmäßig für solche Abfragen keine weiteren Details zurückgeliefert.
- Für die Interpretation der JSON-Datagramme in den Rückgaben gelten die für JSON üblichen Vorgaben zu Syntax und Kodierung.
- In der Anwendungskonfiguration (Datei `application.yml`) vorgenommene Einstellungen (z. B. SSL/TLS, HTTP-Authentifizierung, abweichende Kontext-Pfade etc.) werden im Health-Check berücksichtigt.
- Wenn der ETR-Dienst nicht oder wegen schwerwiegender Fehler nicht erfolgreich gestartet ist, wird keine Antwort zurückgeliefert (der Client erhält i.d.R. ein Timeout).

Damit soll auf einfache, standardisierte Weise die Automatisierung der betrieblichen Zustandsüberwachung mit Hilfe entsprechender Drittanbietersoftware ermöglicht werden.

6.1 Mögliche Ergebnis-Statuswerte

Die Status-Indikation besteht aus einem HTTP-Status-Code im Header und einem JSON-Datagramm im Body der HTTP-Antwort (siehe auch Beispiele unten):

- UP (HTTP-Statuscode 200)
- DOWN (HTTP-Statuscode 503)
- WARNING (HTTP-Statuscode 200)
- UNKNOWN (HTTP-Statuscode 200)

Alle weiteren Statuscodes (z. B. 401 und 403 bei aktivierter Authentifizierung) sind gemäß HTTP-Standard zu verstehen und beziehen sich technisch auf die Ausführung des Health-Checks selbst.

6.2 Beispiele für Anfragen

Systemspezifische Anpassung
Die in den nachfolgenden Beispielen angegebene Basis-URL "http://localhost:8081" muss durch das jeweilige system-spezifische Präfix (Protokoll HTTP vs. HTTPS, konkreter Host- oder DNS-Name, ggf. Kontextpfad) ersetzt werden.

6.2.1 Einfache Abfrage, ob der ETR-Dienst ordnungsgemäß gestartet wurde

URL: http://localhost:8081/actuator/health/ping
Rückgabe (Beispiel):

```
{
  "status": "UP"
}
```

Wenn der ETR-Dienst ordnungsgemäß gestartet ist, wird unter angegebener URL immer dieser Rückgabewert mit HTTP-Statuscode 200 (als Konstante) zurückgegeben.

6.2.2 Spezielle Abfragen zu einzelnen ETR-Komponenten

URL	Art der Prüfung	Mögliche Statuswerte
http://localhost:8081/actuator/health/ericlet	Erreichbarkeit der Eingangsserver (ERiClet): HTTPHead mit Clientzertifikat	UP: mind. ein Eingangsserver ist erreichbar DOWN: Verbindungstest fehlgeschlagen
http://localhost:8081/actuator/health/updateServer	Erreichbarkeit Update-/Download-Server: HTTPHead	UP: erreichbar DOWN: Verbindungstest fehlgeschlagen UNKNOWN: falls das "Netz des Bundes" (NdB) konfiguriert ist
http://localhost:8081/actuator/health/eldas	Erreichbarkeit Datenabholungsserver: HTTPHead	UP: erreichbar DOWN: Verbindungstest fehlgeschlagen UNKNOWN: falls das "Netz des Bundes" (NdB) konfiguriert ist
http://localhost:8081/actuator/health/proxy	Proxy-Server (sofern konfiguriert): akzeptierender Socket	UP: erreichbar DOWN: aktiviert & Verbindungstest fehlgeschlagen UNKNOWN: nicht aktiviert
http://localhost:8081/actuator/health/update	Prüfung auf Aktualität der installierten ETR-Version: Versionsnummer	UP: installierte Version ist aktuell WARNING: installierte Version nicht aktuell UNKNOWN: falls das "Netz des Bundes" (NdB) konfiguriert ist
http://localhost:8081/actuator/health/eingangsDirectory	Eingangsverzeichnis: Verzeichnisprüfung	UP: existiert; schreibbar DOWN: Test fehlgeschlagen
http://localhost:8081/actuator/health/ausgangsDirectory	Ausgangsverzeichnis: Verzeichnisprüfung	UP: existiert; schreibbar DOWN: Test fehlgeschlagen
http://localhost:8081/actuator/health/importDirectory	Import-Verzeichnis: Verzeichnisprüfung	UP: existiert; schreibbar DOWN: Test fehlgeschlagen
http://localhost:8081/actuator/health/datenDirectory	internes Datenverzeichnis: Verzeichnisprüfung	UP: existiert; schreibbar DOWN: Test fehlgeschlagen
http://localhost:8081/actuator/health/archivDirectory ***(ab ETR Version 3.1)***	Verzeichnis, in das (falls aktiviert) die Archivierung erfolgt:	UP: existiert; schreibbar DOWN: Test fehlgeschlagen

URL	Art der Prüfung	Mögliche Statuswerte
	Verzeichnisprüfung	UNKNOWN: falls bei deaktivierter Archivierung kein Archivverzeichnis konfiguriert wurde (z.B. die YAML-Einstellung "etr.archivierung.dateiPfad Schema" leer ist)
http://localhost:8081/actuator/health/auftragsDaemon ***(ab ETR Version 3.3)***	Abfrage des internen Unterbrechungszustands der Hintergrund-Auftragsverarbeitung nach zuvor eingetretener Überlastungssituation des Datenannahmeservers	UP: die Ausführung von Sende- und Bereitstellungsaufträgen normal funktioniert WARNING: die Ausführung von Sende- und Bereitstellungsaufträgen wurde wegen Überlastung des Datenannahmeservers (ERIClet) temporär unterbrochen, wird aber nach Ablauf der konfigurierten (oder vorkonfigurierten) Dauer automatisch fortgesetzt
http://localhost:8081/actuator/health/certificate	Prüfung der konfigurierten Zertifikatsdatei (ELSTER-Zertifikat)	UP: gültiges Zertifkat ist konfiguriert DOWN: Kein Zertifkat konfiguriert, abgelaufen oder nicht entschlüsselbar
http://localhost:8081/actuator/health/berechtigungen	Prüfung auf bestehende Berechtigungen (ELSTER-Konto)	UP: es wurde mindestens eine Berechtigung zum Datenaustausch erteilt DOWN: keine Berechtigungen zum Datenaustausch

6.2.3 Zusammengefasste Übersicht aller Zustandsinformationen

URL: http://localhost:8081/actuator/health/
Rückgabe (Beispiel):


```
{
  "status": "UP",
  "components": {
     "archivDirectory": {
      "status": "UNKNOWN"
    },
    "ausgangsDirectory": {
      "status": "UP"
    },
    "berechtigungen": {
      "status": "UP"
    },
    "certificate": {
      "status": "UP"
    },
    "datenDirectory": {
      "status": "UP"
    },
    "db": {
      "status": "UP"
```

```
    },
    "diskSpace": {
      "status": "UP"
    },
    "eingangsDirectory": {
      "status": "UP"
    },
    "eldas": {
      "status": "UP"
    },
    "ericlet": {
      "status": "UP"
    },
    "importDirectory": {
      "status": "UP"
    },
    "ping": {
      "status": "UP"
    },
    "proxy": {
      "status": "UNKNOWN"
    },
    "update": {
      "status": "UP"
    },
    "updateServer": {
      "status": "UP"
    }
  }
}
```

7 Fortgeschrittene Einstellungen zu Performance und Parallelisierung

Im Folgenden sind erweiterte Konfigurationsschlüssel beschrieben, die das technische Verhalten der Anwendung beeinflussen. Sie können zur Optimierung insbesondere für besonders kleine oder besonders große Installationen angepasst werden.

7.1 Allgemeine Hinweise zu Performance- und Parallelisierungseinstellungen

Fortgeschrittene Konfiguration - Wichtiger Hinweis
Anpassungen der folgenden Performance- und Parallelisierungseinstellungen, die von den Standardwerten im Auslieferungszustand abweichen, können unter Umständen starke (positive oder negative) Auswirkungen auf die Stabilität und das Lastverhalten der Anwendung haben. Sie sollten daher nur von mit dem Betrieb der Anwendung erfahrenen Personal und zunächst unter Beobachtung des Betriebszustands der Anwendung vorgenommen werden, bis in Abhängigkeit der betrieblichen Rahmenbedingungen eine für die konkrete Installation optimierte und stabile Konfiguration gefunden wurde.
Sollten nach Konfigurationsanpassungen Probleme festgestellt werden, die den produktiven Betrieb beeinträchtigen, wird empfohlen, zeitnah die Standardkonfiguration (Auslieferungszustand) wiederherzustellen.

Alle in diesem Kapitel angegebenen Konfigurationsschlüssel sind im Auslieferungszustand in der Datei "application.yml" nicht enthalten, sodass eine implizite "Standardkonfiguration" mit den in den Tabellen angegebenen Werten angewendet wird. Diese Standardkonfiguration ist so gewählt, dass sie für die meisten Anwendungsszenarien einen effizienten, stabilen Betrieb ermöglicht.

Bei Konfigurationsanpassungen, die von dieser Standardkonfiguration abweichen, müssen die Konfigurationsschlüssel mit passender Strukturierung in die entsprechenden Abschnitte der Datei "application.yml" neu eingefügt werden. Die Vorgaben des YAML-1.1-Dateiformats zu Einrückungen und geschützten Sonderzeichen sind dabei zu beachten.

Einige Konfigurationsschlüssel (z.B. mit Präfix "spring" oder "datasource") beziehen sich auf die Konfiguration von Drittanbieterkomponenten (z.B. Datenbank, integrierter Webserver), die in ETR integriert sind oder zusammen mit ETR ausgeliefert werden. Wenn notwendig, wird auf die jeweilige Dokumentation des Herstellers oder Distributors verwiesen.

7.2 Konfiguration des Datenbankverbindungspools

7.2.1 Konfigurationsschlüssel "spring.datasource.hikari.maximum-pool-size"

Definiert die maximale Anzahl der gleichzeitig von ETR verwendeten Datenbankverbindungen. Je höher der eingestellte → Parallelitätsgrad ist, desto mehr gleichzeitige Datenbankverbindungen werden benötigt.

Da ETR die Datenbankverbindungen dynamisch nur bei Bedarf herstellt und in einem "Pool" zwischenspeichert, wird der angegebene Maximalwert nur bei entsprechender Last erreicht. Datenbankverbindungen werden automatisch wieder freigegeben, wenn sie innerhalb einer Zeitspanne (siehe Konfigurationsschlüssel "spring.datasource.hikari.idle-timeout") nicht benötigt werden.

Was bewirkt eine Erhöhung?

- Es werden mehr Ressourcen der Datenbank benötigt.
  - Im Auslieferungszustand verwendet ETR eine integrierte dateibasierte H2-Datenbank. Die Erhöhung des Ressourcenbedarfs der Datenbank erhöht daher den Ressourcenbedarf des laufenden ETR-Diensts.
  - Falls eine externe Datenbank (z.B. Postgres) für ETR konfiguriert ist, muss diese (ressourcentechnisch und lizenztechnisch) in der Lage sein, die benötigte Anzahl an gleichzeitigen Verbindungen für ETR bereitzustellen.

Was bewirkt eine Verminderung?

- Ist der Wert zu niedrig, müssen bei vielen gleichzeitigen Datenbankanfragen (hohe Last) diese im "Pool" ggf. zunächst kurz warten. Dieser Fall führt zu Verzögerungen, beeinträchtigt aber die Stabilität des Systems nicht.
- Sobald der Wartevorgang das Timeout "spring.datasource.hikari.connection-timeout" überschreitet, treten u.U. technische Fehler auf, die zum Abbruch der Verarbeitung führen ("Unable to acquire JDBC Connection"). Ein Wert, der deutlich unter der Standardkonfiguration (Auslieferungszustand) liegt, sollte daher vermieden werden.

7.2.2 Konfigurationsschlüssel "spring.datasource.hikari.connection-timeout"

Steuert die maximale Zeitspanne (Angabe in Millisekunden), die beim Datenbankzugriff auf eine verfügbare Datenbankverbindung aus dem Pool gewartet wird.

Das Warten tritt ein z.B. wenn

- zur Vergrößerung des Verbindungspools eine weitere Verbindung mit einem externen Datenbankserver (z.B. Postgres) hergestellt werden muss
- wegen einer anderen parallelen Verarbeitung datenbankseitig eine angeforderte Ressource temporär belegt/gesperrt ist
- die Datenbank ausgelastet ist

Was bewirkt eine Erhöhung?

- Erhöht die Systemstabilität bei Lastspitzen, da Datenbankoperationen mehr Zeit zur Ausführung erhalten, bevor sie sicherheitshalber abgebrochen werden.
- Im Gegenzug kann dies zu längeren Ausführungszeiten führen, da die Anzahl der Verbindungen im Pool begrenzt ist, was die Wartezeiten für andere gleichzeitige Anfragen/Operationen erhöhen kann.
- Der Wert "spring.datasource.hikari.maximum-pool-size" für die Anzahl Datenbankverbindungen kann ggf. etwas niedriger ausfallen, um Ressourcen der Datenbank zu sparen.

Was bewirkt eine Verminderung?

- Ausführungszeiten der Datenbankoperationen werden tendenziell verringert, was allerdings zum vorzeitigen Abbruch von Datenbankoperationen mit einem technischen Fehler führen kann ("Unable to acquire JDBC Connection"). Ein Wert, der deutlich unter der Standardkonfiguration (Auslieferungszustand) liegt, sollte daher vermieden werden.
- Der Wert "0" bewirkt als Spezialfall, dass Datenbankverbindungen im Verbindungspool dauerhaft belegt werden können. Diese Konfiguration wird nicht empfohlen.

7.2.3 Konfigurationsschlüssel "spring.datasource.hikari.idle-timeout"

Steuert die maximale Zeitspanne (Angabe in Millisekunden), nach der nicht benötige Datenbankverbindungen aus dem "Pool" bis auf Weiteres wieder freigegeben werden.

Was bewirkt eine Erhöhung?

- Latenzen beim Datenbankzugriff können sich verringern, weil Datenbankverbindungen nicht neu hergestellt werden müssen. Dieser Effekt tritt nur kurzzeitig bei Lastwechsel von wenig auf viel Last ein.
- Anwendungsseitig offen gehaltene Datenbankverbindungen belegen Ressourcen der Datenbank, daher werden bei mehr gleichzeitigen Verbindungen dauerhaft tendenziell mehr Datenbankressourcen benötigt.

Was bewirkt eine Verminderung?

- Latenzen beim Datenbankzugriff können sich erhöhen, weil zuvor geschlossene Datenbankverbindungen bei späterer Erhöhung der Last wieder neu hergestellt werden müssen.
- Der Wert "0" bewirkt als Spezialfall, dass temporär nicht benötigte Datenbankverbindungen nie freigegeben werden. Diese Konfiguration wird nicht empfohlen.

7.2.4 Wertebereich-Matrix

Konfigurationsschlüssel	min.	Standardkonfiguration (Auslieferungszustand)	max.	Link zur Dokumentation des Herstellers/Distributors
spring.datasource.hikari.maximum-pool-size[1]	1	**30**	beliebig	→ HikariCP (Configuration)
spring.datasource.hikari.connection-timeout	0	**30000**	beliebig	
spring.datasource.hikari.idle-timeout	0	**30000**	beliebig	

[1]) Der angegebene Wert muss vom Datenbankserver unterstützt werden, falls eine externe Datenbank (z.B. Postgres) verwendet wird.

7.3 Konfiguration des integrierten HTTP-Webservers (Web-GUI, REST API)

7.3.1 Konfigurationsschlüssel "server.tomcat.threads.min-spare"

Definiert die Mindestgröße des Thread-Pools für eingehende HTTP-Requests (Web-GUI und REST API) des in ETR integrierten Apache-Tomcat-Webservers. Mindestens die angegebene Anzahl an Threads wird auch in Niedriglastsituationen untätig im Pool gehalten, wenn nicht ausreichend eingehende HTTP-Request eintreffen.

Sollte kleiner oder gleich "server.tomcat.threads.max" sein.

7.3.2 Konfigurationsschlüssel "server.tomcat.threads.max"

Definiert die maximale Größe des Thread-Pools für eingehende HTTP-Requests (Web-GUI und REST API) des in ETR integrierten Apache-Tomcat-Webservers.

Die tatsächliche Größe des Thread-Pools wird von ETR zur Laufzeit automatisch angepasst: Bis zum angegebenen Maximalwert viele Threads werden erzeugt, sobald sie benötigt werden.

Abgrenzung: Die Verarbeitung von größeren und potenziell langlaufenden Downloads erfolgt stattdessen "asynchron" in einem → eigenen Thread-Pool.

Sollte größer oder gleich "server.tomcat.threads.min-spare" sein.

7.3.3 Konfigurationsschlüssel "server.tomcat.accept-count"

Maximale Warteschlangenlänge für eingehende HTTP-Requests (Web-GUI und REST API), für den Fall, dass der Thread-Pool des integrierten Webservers bereits voll ausgelastet ist.

Überzählige HTTP-Requests, die keinen Platz in der Warteschlange finden (=Überlastsituation), werden aus Sicherheitsgründen auf Socket-Ebene abgewiesen.

7.3.4 Konfigurationsschlüssel "etr.http-server.async.core-pool-size"

Definiert die Mindestgröße des Thread-Pools, der zur Verarbeitung von "asynchronen" HTTP-Anfragen dient. ETR behandelt per Web-GUI oder REST API eingehende HTTP-Anfragen nur "asynchron" bei potenziell lang dauernden Download-Vorgängen d.h.

- Herunterladen von Eingangsdateien im Webbrowser (Web-GUI)
- Herunterladen von Eingangsdateien per REST API (URL-Suffix "/rest/abholauftrag/*{auftragId}*/download/*{downloadId}*")

Die tatsächliche Größe des Thread-Pools wird von ETR zur Laufzeit automatisch angepasst: Bis zum angegebenen Wert viele Threads werden erzeugt, sobald sie benötigt werden.

Sollte kleiner oder gleich "etr.http-server.async.max-pool-size" sein.

7.3.5 Konfigurationsschlüssel "etr.http-server.async.max-pool-size"

Definiert die Maximalgröße des Thread-Pools, der zur Verarbeitung von "asynchronen" HTTP-Anfragen dient.

Treffen über den unter "etr.http-server.async.core-pool-size" angegebenen Wert hinaus weiterhin mehr Anfragen ein als Threads vorhanden sind, wird zunächst eine Warteschlange gebildet. Sobald die Warteschlange voll ist d.h. "etr.http-server.async.queue-capacity" viele Einträge umfasst, werden zur Beseitigung der Lastspitze weitere Threads erzeugt (bis der angegebene Maximalwert "etr.http-server.async.maxPoolSize" erreicht ist). Diese Threads werden später automatisch wieder freigegeben, sobald die Lastspitze endet.

Falls "etr.http-server.async.core-pool-size" und "etr.http-server.async.max-pool-size" gleich gesetzt werden, werden keine zusätzlichen Threads für Lastspitzen erzeugt.

Sollte größer oder gleich "etr.http-server.async.core-pool-size" sein.

7.3.6 Konfigurationsschlüssel "etr.http-server.async.queue-capacity"

Definiert die Größe der Warteschlange, die zur Verarbeitung von "asynchronen" HTTP-Anfragen dient, für den Fall, dass alle im Pool vorhandenen Threads ("etr.http-server.async.core-pool-size" viele) bereits Anfragen bearbeiten und weitere Anfragen eintreffen. Erst wenn die Anzahl der wartenden Anfragen in der Warteschlange den angegebenen Maximalwert überschreitet, werden zur Bewältigung von Lastspitzen temporär weitere Threads automatisch erzeugt, bis auch "etr.http-server.async.max-pool-size" erreicht ist.

7.3.7 Konfigurationsschlüssel "spring.mvc.async.request-timeout"

Steuert die Zeitdauer (Timeout), die maximal vergehen darf, bevor "asynchrone" HTTP-Anfragen aus Sicherheitsgründen abgebrochen werden.

Die Zeitspanne sollte so gewählt sein, dass auch das Zurücksenden der größten von ETR abgeholten Eingangsdateien an einen aufrufenden Webbrowser oder REST-Client in Abhängigkeit der verfügbaren Netzwerkbandbreite immer ordnungsgemäß möglich ist.

Die Angabe erfolgt in Millisekunden (z.B. "1200000" = 20 Minuten) oder mit den üblichen Einheiten als Suffix ("s" = Sekunden, "m" = Minuten", "h" = Stunden, "d" = Tage).

Der Wert "0" definiert als Spezialfall, dass kein Abbruch nach einem Timeout erfolgt. Diese Konfiguration wird aus Sicherheitsgründen nicht empfohlen.

7.3.8 Wertebereich-Matrix

Konfigurationsschlüssel	min.	Standardkonfiguration (Auslieferungszustand)	max.	Link zur Dokumentation des Herstellers/Distributors
server.tomcat.threads.max	0	**200**	beliebig	→ Spring Boot / Common Application Properties
server.tomcat.threads.min-spare	0	**10**	beliebig	→ Spring Boot / Common Application Properties
server.tomcat.accept-count	0	**100**	beliebig	→ Spring Boot / Common Application Properties
etr.http-server.async.core-pool-size	1	**10**	beliebig	
etr.http-server.async.max-pool-size	1	**20**	beliebig	
etr.http-server.async.queue-capacity	1	**200**	beliebig	
spring.mvc.async.request-timeout	0	**1h**	beliebig	→ Spring Boot / Common Application Properties

7.4 Konfiguration des integrierten HTTP-Clients (Zugriff auf Backendsysteme)

7.4.1 Konfigurationsschlüssel "etr.http-client.max-total-connections"

Anzahl gleichzeitig geöffneter HTTP-Client-Verbindungen.

Da ETR die HTTP-Verbindungen zu Backend-Systemen dynamisch nur bei Bedarf herstellt und in einem "Pool" zwischenspeichert, wird der angegebene Maximalwert nur bei entsprechender Last erreicht.

7.4.2 Konfigurationsschlüssel "etr.http-client.max-connections-per-host"

Anzahl gleichzeitig geöffneter HTTP-Client-Verbindungen zum selben Ziel-Server.

Dient dazu, die Last der Netzwerkinfrastruktur bzw. Gegenstelle zu steuern und ggf. zu beschränken.

7.4.3 Konfigurationsschlüssel "etr.http-client.connect-timeout"

Zeitvorgabe (Timeout), die maximal vergehen darf, bis ETR (als Client) eine HTTP-Verbindung zu einem Ziel-Server herstellen kann. Dies umfasst bei gesicherten Verbindungen ggf. auch die Zeit zur Bereitstellung der benötigten SSL-/TLS-Transportsicherheit. Angabe in Sekunden.

7.4.4 Konfigurationsschlüssel "etr.http-client.response-timeout"

Zeitvorgabe (Timeout), die maximal vergehen darf, bis ETR (als Client) auf eine Anfrage über eine bereits hergestellte HTTP-Verbindung eine Antwort vom Ziel-Server erhält. Angabe in Sekunden.

7.4.5 Wertebereich-Matrix

Konfigurationsschlüssel	min.	Standardkonfiguration (Auslieferungszustand)	max.
etr.http-client.max-total-connections	1	**60**	5000
etr.http-client.max-connections-per-host	1	**6**	1000
etr.http-client.connect-timeout	1	**180**	beliebig
etr.httpClient.response-timeout	1	**1800**	beliebig

7.5 Parallelitätsgrad der Datentransfers

Mit Hilfe der folgenden, **gegenseitig voneinander abhängigen** Konfigurationsschlüssel kann gesteuert werden, ob und wie viele Datentransfers ETR gleichzeitig (parallel) im Hintergrund ausführt. Jeder Konfigurationsschlüssel definiert den Parallelitätsgrad auf einer von 3 Ebenen.

Der effektive Nutzen der Parallelisierung ist stark von der Menge und Struktur der übertragenen Daten abhängig. Vorteile entstehen bei der Verarbeitung von vielen Aufträgen (hohe Last) in gut mit Systemressourcen ausgestatteten Umgebungen (leistungsfähige Mehrkern-CPUs, gute Netzwerkanbindung an die ELSTER-Backend-Systeme). Parallelität hat keine signifikanten Vorteile bei geringer Last, wenn über ETR nur sporadisch Datentransfers stattfinden.

Was bewirkt eine Erhöhung?

- Parallelisierung kann die Effizienz der Datentransfers im Durchschnitt steigern.
- Je höher der Wert, desto mehr Systemressourcen (CPU, freier Hauptspeicher, Netzwerkbandbreite) werden für die Datenübertragung benötigt. Ggf. müssen weitere Parameter angepasst werden, damit das System unter diesen Bedingungen stabil ist.

Was bewirkt eine Verminderung?

- Bei Wert "1" wird die Parallelisierung auf der jeweiligen Ebene deaktiviert. Die Verarbeitung erfolgt dann sequenziell. Im Fall von technischen Problemen bedeutet dieser Modus eine tendenziell höhere Systemstabilität.
- Je kleiner der Wert, desto länger dauern tendenziell die Datenübertragungen in Summe.

7.5.1 Konfigurationsschlüssel "etr.daemon.parallelism"

Gibt an, wie viele Aufträge parallel verarbeitet werden können.

Ab ETR 25.01 findet die Parallelisierung weitgehend auch über die verschiedenen Auftragsarten (Abhol-, Sende, Bereitstellungsaufträge) übergreifend statt.

7.5.2 Konfigurationsschlüssel "etr.daemon.downloadParallelism"

Betrifft nur **Abholaufträge** (Einzel- oder Dauerabholaufträge inkl. Abholung eingehender Dokumente im Posteingang).

Gibt an, wie viele Downloads parallel verarbeitet werden können. Der Wert wirkt sich insbesondere auf Daueraufträge aus, bei denen viele Downloads innerhalb einer einzigen Ausführung stattfinden.

Empfehlung, um unnötige Wartezeiten während der Verarbeitung zu vermeiden:

- sollte mindestens so groß wie "*etr.daemon.parallelism*" sein.
- sollte höchstens so groß wie "*etr.otter.max-parallel-transfers*" sein

7.5.3 Konfigurationsschlüssel "etr.otter.max-parallel-transfers"

Gibt an, wie viele Dokumente von/zum ELSTER-Objektspeicher (OTTER) parallel übertragen werden können.

Dies betrifft nur Aufträge zu Auftragsarten, die aus technischer Sicht den ELSTER-Objektspeicher (OTTER) zur Übertragung nutzen:

- Zentrales Unternehmenspostfach: **Bereitstellungsaufträge mit Anhängen ab 10 MiByte** (ausgehende Dokumente)
- Zentrales Unternehmenspostfach: **Posteingang** (eingehende Dokumente)

In OTTER gespeichert werden nur die "Anhänge" (Dokumentteile) der Aufträge. Der angegebene Wert wirkt sich daher nur aus, wenn Aufträge mehrere Anhänge aufweisen oder mehrere Aufträge mit entsprechenden Anhängen gleichzeitig abgearbeitet werden.

Zu berücksichtigen ist, dass jede OTTER-Anfrage jeweils eine HTTP-Verbindung benötigt (siehe Konfigurationsschlüssel "etr.http-client.max-total-connections" und "etr.http-client.max-connections-per-host"). Die Einstellungen für die maximale Anzahl der gleichzeitigen HTTP-Verbindungen sollten beide größer oder gleich dem angegebenen Wert sein, um Blockierungen bei der Ausführung der OTTER-Anfragen zu vermeiden.

7.5.4 Wertebereich-Matrix

Konfigurationsschlüssel	min.	Standardkonfiguration (Auslieferungszustand)	max.
etr.daemon.parallelism	1	**2**	8
etr.daemon.downloadParallelism	1	entspricht "*etr.otter.max-parallel-transfers*"	20
etr.otter.max-parallel-transfers	1	**2**	20