

Inhaltsverzeichnis	1
Kernmodul Komponentenverwaltung	2
Installation von Modulen	2
Architektur von SuperX-Modulen	2
Modulscripte im Kernmodul	2
Installation eines Moduls: Allgemeines Vorgehen	3
Allgemeines zum Laden	3
Einrichtung der Entladescrpte	3
Entladen unter Windows	4
Dateitransfer beim Push-Verfahren	4
Entfernen der Passwor eingabe unter Unix	4
Entfernen der Passwor eingabe unter Windows	5
Einrichtung von SFTP (Server)	5
Einrichtung von SFTP	5
Einrichtung von SFTP unter Windows	6
Java-Client zum Entladen von Quell-Datenbanken	7
Update eines Moduls: Allgemeines Vorgehen	8
Modulupdate in mandantenfähigen Installationen	8
Format der Unload Dateien CSV	9
Upgrade eines Moduls: Allgemeines Vorgehen	9
Hochschulspezifische Anpassung eines Moduls	9
Dateibasierte hochschulspezifische Anpassung	9
Datenbankbasierte hochschulspezifische Laderegeln	9
Entfernen eines Moduls	10

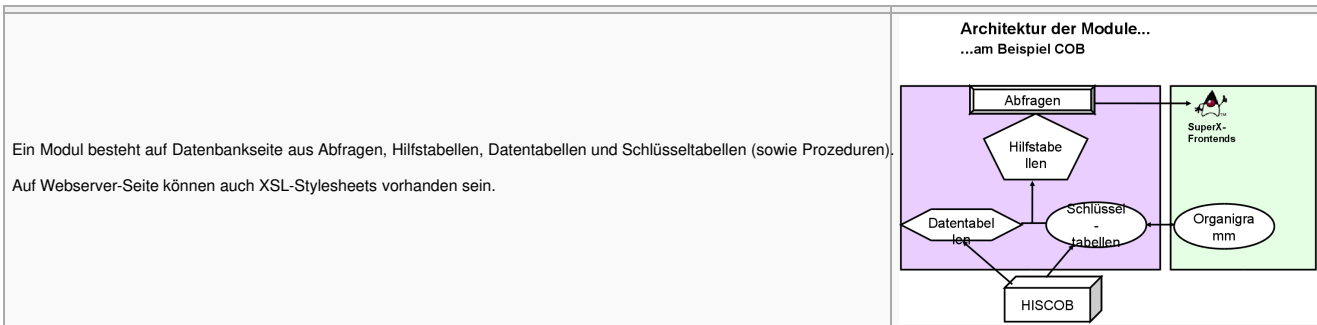
Kernmodul Komponentenverwaltung

Installation von Modulen

Das Kernmodul enthält außer Administrationsabfragen und Tabellen keinerlei Inhalte. Die Inhalte werden in Form von Modulen hinzugefügt. Dazu gibt es vorgefertigte [Installationscripte](#).

Architektur von SuperX-Modulen

Die folgende Abbildung zeigt die Architektur von Modulen am Beispiel vom HISCOB-Modul:



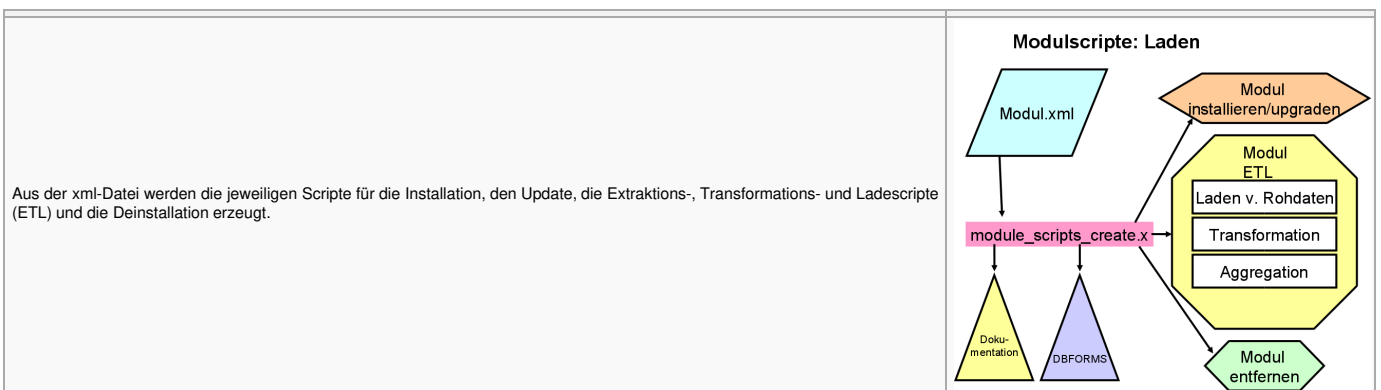
Die Abbildung zeigt, dass ein Modul eigene Komponenten nutzt, aber auch auf Teile des Kernmoduls zugreift, z.B. das Organigramm - dies macht SuperX zu einem integrierten System. Neben dem Organigramm sind alle anderen Komponenten des Kernmoduls natürlich betroffen, z.B. Themenbaum, Userrechte.

Die Ordnerstruktur eines Moduls spiegelt die Komponenten des Systems wieder. Es gibt je ein Verzeichnis für datentabellen, schluessestabellen und hilfstabellen.

Die Installation eines Moduls ist in der Dokumentation des jeweiligen Moduls näher beschrieben. Module, die auf dem Kernmodul 2.1 oder höher basieren, haben einen einheitlichen Aufbau.

Modulscripte im Kernmodul

Seit Version 2.1 werden die Datenbankschemata und Scripte der Module in einem einheitlichen Format zusammengestellt und in einer Datei `$$SUPERX_DIR/db/module/$MODULNAME/conf/$MODULNAME.xml` gespeichert. Das XML-Format hat den Vorteil, dass die Scripte dynamisch für Postgres und Informix erzeugt werden können, und dass die Scripte vereinheitlicht werden. Aus dieser Datei werden die Scripte erzeugt, die das Modul jeweils für Postgres und Informix installieren / updaten / aktualisieren / überprüfen und entfernen. Die folgende Abbildung zeigt das Vorgehen:



Die Modul-Scripte liegen als Shellscrip te im Verzeichnis `$$SUPERX_DIR/db/bin`, und sind an [anderer Stelle](#) im Detail erläutert.

module_install.x -name- -pfad- z.B.: module_install.x sos \$\$SOS_PFAD	Installiert ein Modul, d.h. erzeugt die Tabellen, Views, Funktionen und Abfragen. Die Abfragen werden in den Themenbaum eingehängt.
module_etl.x -name- -pfad- z.B.: module_etl.x sos \$\$SOS_PFAD	Aktualisiert das Modul, d.h. lädt die Rohdaten, Transformiert sie, und aggregiert die Hilfstabellen. Fehler werden in eine Log-Datei geschrieben, diese kann dann per Mail an einen Admin versandt werden. Wenn das Script erfolgreich durchläuft, wird die Tabelle systeminfo aktualisiert. Das Script wird nicht direkt als Cronjob aufgerufen, sondern von einem Shellscrip t, das die Umgebungsvariablen und Parameter setzt, z.B. cob_update.x.
module_drop.x -name- -pfad- z.B.: module_drop.x sos \$\$SOS_PFAD	Entfernt das Modul bzw. die Tabellen, Views, Funktionen und Abfragen.
module_scripts_create.x -name- -pfad- - Datenbanksystem(optional, INFORMIX, POSTGRES)- -Versionsnr.-	Erzeugt via XSL-Transformation die jeweiligen Scripte, die in den obigen Scripten aufgerufen werden, z.B. bau_install_ids.sql für das Installationscript des Baumoduls beim Informix Dynamic Server.

Neben den operativen Scripten erzeugt `module_scripts_create.x` auch html-Dateien zur Dokumentation eines Moduls in

`$SUPERX_DIR/db/module/-Modulname-/conf/-Modulname-.html`

(auch als rtf-Datei zu Einbindung in Modul-Dokumentationen) sowie zur Schnittstelle in

`$SUPERX_DIR/db/module/-Modulname-/rohdaten/-Modulname-_unload.html`

Darüber hinaus werden auch DBForms-Formulare erzeugt.

Installation eines Moduls: Allgemeines Vorgehen

Das Vorgehen bei der Installation eines Moduls ist standardisiert. Im folgenden eine Kurzbeschreibung, weiter unten finden Sie das Vorgehen am Beispiel des ZUL-Moduls im Detail.

- Entpacken Sie das Modul in `$SUPERX_DIR`
- Erweiterung der Umgebung in der Datei `$SUPERX_DIR/db/bin/SQL_ENV`: fügen Sie den Inhalt der jew. Beispieldatei `SQL_ENV_-Modulname-.sam` in der `SQL_ENV` an, wenn noch nicht vorhanden, und ändern Sie ggf. Email-Adressen für log- und Fehlermails.
- Entladen der Rohdaten; auch hier müssen vorher Umgebungsvariablen zum Vorsystem angepasst werden

(Datei `$-Modulname-_LOAD_PFAD/-Modulname-_ENV`), hier liegt ebenfalls eine *.sam-Datei vor

- Kopieren der Rohdaten nach `-Modulpfad/-rohdaten`. Neuere SuperX-Module haben dafür vorgefertigte Scripte mit dem Namen `-Modulname-_copy.x` (z.B. `zul_copy.x`)
- Installieren Sie das Modul mit `-Modulname-_erzeugen.x`, z.B. mit `zul_modul_erzeugen.x`
- Wenn die Installation erfolgreich war, können Sie das Modul aktualisieren mit

`-Modulname-_update.x` (ggf. mit Parametern)

z.B. mit

`zul__update.x`

d.h. die ETL-Prozesse werden gestartet (s.u.).

- Wenn das Modul erfolgreich aktualisiert ist, wird eine Prüfprozedur gestartet, die die Daten plausibilisiert. Fehler und Warnungen finden Sie in der Datei `$-Modulname-_ERRORDAT`.
- Starten Sie Tomcat neu. Wenn Tomcat auf einem separaten Server installiert ist, müssen Sie vorher die Datei `$SUPERX_DIR/webserver/tomcat/webapps/superx/WEB-INF/dbforms-config.xml` dorthin übertragen.
- Schritt 7 wird bei jedem SuperX-Update wiederholt. Nun muss der Entladerrhythmus geplant werden, und die Cronjobs werden eingerichtet. Es gibt eine Musterdatei `$SUPERX_DIR/db/module/-MODULNAME/-MODUL-_update_cron.x.sam`, benennen sie diese um nach `-MODUL-_update_cron.x` und fügen Sie das Script in die crontab ein.

Allgemeines zum Laden

Zunächst müssen die Rohdaten aus dem Vorsystem entladen werden.

Für das Entladen gibt es ferner zwei Modi: Das "Pull"-Verfahren und das "Push"-Verfahren.

- Beim **"Pull"**-Verfahren wird einer Benutzerkennung auf dem SuperX-Rechner Zugriffsrecht auf die SOSPOS-Datenbank gegeben, und die Daten werden via TCP/IP aus dem Basissystem entladen. Bei APP / HISinOne als Quellsystem ist "Pull" das Standardvorgehen.
- Beim **"Push"**-Verfahren werden die Entladescripte auf den SOSPOS-Rechner kopiert und dort von einer Benutzerkennung auf dem SOSPOS-Rechner ausgeführt. Die Rohdaten müssen dann auf den SuperX-Rechner kopiert werden. Dieses Verfahren klappt bei Informix unter Unix problemlos, bei Entladen aus Postgres müsste das komplette SuperX-Kernmodul installiert werden.

Am einfachsten ist immer das "Pull"-Verfahren, das mit fast allen Quellsystemen funktioniert und wenig Konfiguration auf dem Quellsystem erfordert. Aufgrund von Sicherheitsvorkehrungen oder Netz-Infrastrukturen wählen aber viele Hochschulen das "Push"-Verfahren. Da derzeit Informix / Unix die gängigste Plattform an Hochschulen ist, ist dies auch kein Problem.

Einrichtung der Entladescripte

Im Push-Verfahren bzw. unter SuperX können Sie mitgelieferte Shellscripته nutzen. Das folgende Beispiel des ZUL-Moduls zeigt das Vorgehen:

Die Entladescripte liegen im Verzeichnis `$SUPERX_DIR/db/module/zul/rohdaten` und lauten je nach Systemversion:

`zul_unload.x`

Die Scripte laufen nur, wenn die entsprechenden Umgebungsvariablen in der Datei `ZUL_ENV` (im gleichen Verzeichnis, ein Muster liegt vor in `ZUL_ENV.sam`) korrekt gesetzt sind, benennen Sie die Musterdatei um nach `ZUL_ENV` und tragen die richtigen Umgebungsvariablen ein, z.B. den Pfad für `$INFORMIXDIR`.

ZUL_ENV Die Umgebung für Entladescripte aus ZUL-GX(Auszug)

```
##Pfad für Entladedaten:
ZUL_PFAD=.; export ZUL_PFAD
##hier muss Unterverzeichnis unl existieren!
```

In der `ZUL_ENV` müssen außerdem folgende Umgebungsvariablen gesetzt werden (defaults sind bereits vorgelegt, aber hier und da müssen Sie sicher ran):

Variable	Erläuterung
	Nur für Informix gelten:
INFORMIXDIR	Home-Verzeichnis von Informix
INFORMIXSERVER	Name des Informixservers
ONCONFIG	Name der onconfig, wenn auf dem SOS-Rechner mehrere Informix-Instanzen laufen
CLIENT_LOCALE	Sprachumgebung (wichtig fürs Entladen von Datumsformaten)
SERVER_LOCALE	dito

Variable	Erläuterung
	Nur für Postgres gelten:
PGDATESTYLE	Datumsformat "German"
PGPORT	Port vom Postgres-Server, standardmäßig 5432
PGHOST	Hostname oder IP-Adresse vom Postgres-Server
PGUSER	Benutzerkennung für Postgres-Server (nur Datenbank, nicht Betriebssystem)
PGPATH	Installationsverzeichnis von Postgres, z.B. <code>/usr/local/pgsql</code>
DB_PROPERTIES	Pfad zur <code>db-zul.properties</code> -Datei mit den Zugangsparametern für SOSPOS unter Postgres
LOGGING_PROPERTIES	Pfad zur Steuerungsdatei mit den Parametern für das Logging beim Entladen, voreingestellt auf <code>./logging.properties</code> . Normalerweise brauchen Sie hier nichts ändern, wenn beim Entladen Probleme auftauchen, kann man den Level von SEVERE auf INFO oder FINEST ändern, dann werden die konkreten SQLs geloggt. Aber Achtung: wenn keine Fehler mehr auftreten, müssen Sie den Level wieder auf SEVERE ändern, sonst kommen Schlüsselworte in die Logdatei <code>zul_unload.err</code> , die

dann bei der Übernahme nach SuperX fälschlicherweise zu Fehlermeldungen führen.

Unter Postgres muss für das "Pull"-Verfahren beim Entladen die Datenbankverbindung in der Datei db-zul.properties eingetragen werden (Muster für Postgres liegt bei in db-zul_pg.properties). Dazu laden Sie einmal die Datei ZUL_ENV mit den obigen Parameter, starten den SuperX-Propadmin (siehe Administrationshandbuch Kernmodul) und richten die Verbindung zum SOSPOS-Server ein. Das Kennwort wird verschlüsselt gespeichert. Danach sind die Entladescripte für Postgres ausführbar.

Hinweis: Anders als Informix hat Postgres eine eigene, vom Basissystem unabhängige Benutzerverwaltung. Daher brauchen Sie den User, den Sie zum Entladen aus Postgres nutzen, nicht auf dem SuperX- oder SOSPOS-Rechner auf Betriebssystem-Ebene einrichten. Sie können also z.B. auf dem SuperX-Rechner zum Entladen aus SOSPOS die Kennung sospos des Postgres-Rechners verwenden. Oder Sie richten in der SOSPOS -Datenbank den Benutzer SuperX ein und geben ihm Leserecht auf die Tabellen sowie das Recht, Tabellen und Stored Procedures anzulegen.

Variable	Erläuterung
	Für alle Plattformen gelten folgende Variablen:
ERRORMAIL	An wen solle eine Logmail verschickt werden, wenn das Entladen nicht geklappt hat? (nur Unix).
LOGMAIL	An wen soll immer eine Logmail verschickt werden
MAILPROG	Platz zum ausführbaren Mailprogramm unter Unix, Vorbelegung ist "mail", manche Unixe haben aber auch "mutt" oder "mailx".
	Wenn die Rohdaten beim Push-Verfahren nach dem Entladen vom ZUL-Rechner auf den SuperX-Rechner kopiert werden sollen, dann werden für das Script zul_copy.x folgende Umgebungsvariablen benötigt:
COPY_METHOD	Programm, das die Dateien kopiert; rsync und scp sind wählbar.
REMOTE_DIR	Verzeichnis, in das die Rohdaten auf dem SuperX-Rechner kopiert werden sollen, in der Regel ist dies "/home/superx/db/module/zul/rohdaten"
REMOTE_USER	Der Unix-Username auf dem SuperX-Rechner, in der Regel "superx".
REMOTE_HOST	Der Rechnername bzw. die IP-Nr. des SuperX-Rechners.

Dann starten Sie das Script zul_unload.x. Wenn es gelaufen ist, müssten die Dateien im unl-Verzeichnis stehen. Prüfen Sie dann bitte, ob dort Dateien mit 0 bytes stehen. Die Logdatei heisst zul_unload.err.

Entladen unter Windows

Für das Entladen unter Windows gibt es Muster-Dateien im jew. Rohdaten-Verzeichnis, z.B. \$SOS_PFAD/rohdaten/sos_unload.bat. Die Einrichtung geschieht dann in [Java](#). Ans Ende der Datei können Sie noch schreiben:

```
date /T >superx.datum
```

Dann wird auch die Datumsdatei geschrieben.

Dateitransfer beim Push-Verfahren

Wenn Sie das Verzeichnis nicht gemounted haben, müssen das Verzeichnis unl, die zul_unload.err und die superx.datum dann in das Verzeichnis \$ZUL_LOAD_PFAD auf dem SuperX-Rechner kopiert werden, ein Script dafür liegt ebenfalls bei (zul_copy.x). Das Entladedatum wird danach in der Textdatei \$ZUL_LOAD_PFAD/superx.datum gespeichert; wenn das Script einen Fehler findet, dann wird das vorherige Datum (in der Datei superx.datum.alt) gesetzt.

Für den Transfer der Rohdaten beim Push-Verfahren wird in SuperX die dateibasierte Schnittstelle genutzt. Unter UNIX läßt sich dieser Transfer vollends automatisieren, indem die Programme scp oder rsync auf der Basis des OpenSSH-Pakets genutzt werden. Beide setzen auf das ssh-Protokoll 2 auf und stellen somit einen verschlüsselten Dateitransfer sicher. Auch [sftp](#) ist möglich.

In den jeweiligen Modulen wird im Verzeichnis rohdaten eine Beispieldatei mit dem Namen -MODULNAME- _ENV.sam ausgeliefert, die Sie umbenennen können nach -MODULNAME- _ENV. Darin werden am Ende der Datei die Parameter zum Kopieren festgelegt, also die Userkennung REMOTE_USER, der Hostname REMOTE_HOST, und die Methode des Kopierens (COPY_METHOD) sowie die jeweiligen Zielpfade. Diese Umgebungsvariablen werden von dem jeweiligen Script -modulname-_copy.x benutzt.

Entfernen der Passwortheingabe unter Unix

Damit die Passwortheingabe unter Unix entfällt, muss man wie folgt vorgehen:

Loggen Sie sich zunächst testweise einmal ein. Wenn Sie z.B. vom COB-Server auf den SuperX-Server kopieren wollen, loggen Sie sich als user cob auf cobhost ein mit

```
ssh superx@superxhost
```

Beim ersten Mal müssen Sie die Sicherheitsabfrage mit "yes" bestätigen.

Erzeugen Sie auf dem Quellrechner einen öffentlichen Schlüssel mittels ssh-keygen -t rsa, wobei man eine leere Passphrase vergibt (Achtung: mögliche Sicherheitslücke!). Der öffentliche Teil dieses Schlüssels (~/.ssh/id_rsa.pub) muss auf dem Zielrechner in die Datei ~/.ssh/authorized_keys eingefügt werden, ggf. muss die Datei neu erzeugt werden.

Wenn z.B. auf dem COB-Server unter der Kennung cob ein Key wie folgt erzeugt wurde:

```
Beispielintrag eines Public Keys:
ssh-rsa AAAAB3Nza...viele kryptische Zeichen...pg6VkcC= cob@cobhost
```

Dann wird genau diese Zeile in der Datei /home/superx/.ssh/authorized_keys angefügt (die Datei kann mehrere PublicKeys enthalten, ein Eintrag pro Absatz).

Achten Sie auch auf Dateirechte: Die Verzeichnisse und Dateien sollten keine Schreibrechte für Gruppen haben. Im Zweifelsfall z.B. für den user superx:

```
chmod 700 /home/superx/.ssh
chmod 700 /home/superx/.ssh
chmod 600 /home/superx/.ssh/authorized_keys
```

Danach sollte z.B. der Login vom cobhost als user cob mit ssh superx@superxhost ohne Passwortheingabe klappen. Wenn nicht, schalten Sie das Logging mit ssh -v superx@superxhost ein. Eine Möglichkeit ist, dass die PublicKey-Authentifizierung in der Konfigurationsdatei des SSHD (normal /etc/ssh/ssh_config) abgeschaltet ist. Weitere Diagnosen liefert die Datei /var/log/messages.

Sie können außerdem noch einschränken, von welchem Host die obige Authentifizierung ermöglicht wird. Dazu setzen Sie den Parameter "from="uni-xy.de" davor, z.B.

Einschränkung "from"	from="*uni-xy.de" ssh-rsa AAAAB3Nza...viele kryptische
in authorized_keys	Zeichen...pg6VkcC= cob@cobhost

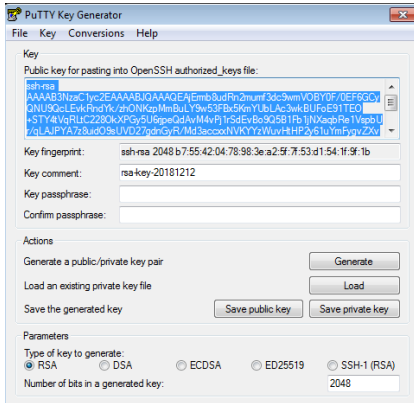
Wenn Sie die Kopiermethode scp benutzen, und die obige "authorized_keys"-Methode mit PublicKey nicht nutzen wollen, können Sie auch mit Private Keys arbeiten (siehe SSH-Doku). Dazu können sie in der *_ENV-Datei in dem Parameter SCP_OPTS den Verweis auf den private Key setzen.

SCP_OPTS in *_ENV:	SCP_OPTS="-p -B -i /home/cob/.ssh/superx_key"
Beispiel COB_ENV	export SCP_OPTS

Entfernen der Passwordeingabe unter Windows

Wenn Sie die ssh-Shell putty aus dem Installationspaket nutzen, haben Sie im Installationsordner auch die Anwendung "puttyGen.exe". Wenn Sie putty als reine Executable nutzen, laden Sie die Anwendung separat herunter. [F](#)

- Folgen Sie der Anleitung auf der Seite http://winscp.net/eng/docs/ui_puttygen, um den SSH2-DSA oder RSA Schlüssel zu erzeugen. Für den Schlüssel bitte kein Passwort vergeben.
- Den privaten Schlüssel geben Sie in Putty bzw. Winscp im Menü "Session" im Feld "Private Key File" an.
- Der Button "Save public key" ist nicht für OPENSSH unter Linux als Zielsystem geeignet. Im folgenden Screenshot sehen Sie den markierten Bereich, den Sie kopieren müssen:



- Kopieren Sie Ihren Public Key, den Sie oben mit PuttyGen erzeugt haben, zur Datei "authorized_keys" und kopieren Sie diese in das Verzeichnis ~/.ssh/
- Bei erneutem Login mit Putty oder Winscp sollte die Passwortabfrage entfallen.

Einrichtung von SFTP (Server)

Beim Push-Verfahren wird, wenn mit scp oder rsync kopiert wird, eine Login-Shell vorausgesetzt. Wenn dies aus Sicherheitsgründen nicht gewünscht ist bzw. wg. Einsatz von Windows nicht möglich ist, können Sie auch sftp nutzen, dies wird in modernen ssh-Servern mitgeliefert und bietet ebenfalls verschlüsselten Datentransfer. Zur Einrichtung des Servers:

Der SSH-Dienst wird wie folgt konfiguriert (am Beispiel Ubuntu Linux 22.04 LTS) mit dem SFTP-Verzeichnis /home/sftp/-benutzername:

In der Datei /etc/ssh/sshd_config die Zeile auskommentieren:

```
#Subsystem sftp /usr/lib/openssh/sftp-server
```

und stattdessen:

```
Subsystem sftp internal-sftp
Match Group sftponly
ChrootDirectory /home/sftp/%u
ForceCommand internal-sftp
AllowTcpForwarding no
```

Dann legen Sie die Gruppe "sftponly" an:

```
groupadd sftponly
und dann den Unix-User an, und geben ihm die Gruppe "sftponly". Hier ein Script:
#!/bin/sh
SFTPUSENAME=$1
mkdir -p /home/sftp/$SFTPUSENAME
useradd -b /home/sftp -s /bin/false -G sftponly $SFTPUSENAME
chown root:root /home/sftp/$SFTPUSENAME
chmod 755 /home/sftp/$SFTPUSENAME
mkdir -p /home/sftp/$SFTPUSENAME/incoming
chown $SFTPUSENAME:sftponly /home/sftp/$SFTPUSENAME/incoming
```

Danach legen Sie mit

```
passwd $SFTPUSENAME
```

ein Passwort fest.

Der User kann sich dann nicht mehr mit ssh einloggen, nur noch mit sftp, und landet beim Login in /home/\$KENNUNG, in einem "chroot-Käfig", und der User kann mit SFTP in den Ordner "incoming" schreiben. Zum Testen geben Sie ein:

```
sftp $KENNUNG@-Host-
```

Wenn beim SFTP Login die Meldung kommt:

```
client_loop: send disconnect: Broken pipe
```

müssen Sie darauf achten dass das Home-Verzeichnis des Users nicht dem User gehört, sondern root.

Wenn der Login klappt, können Sie nach dem [oben](#) beschriebenen Verfahren die Passwordeingabe durch PublicKey-Authentifizierung ersetzen.

Einrichtung von SFTP

Für das Kopieren der Rohdaten können Sie im Kopierscript "-modulname-_copy.x" auch sftp-Kommandos nutzen. Sie müssen nur in der ENV-Datei die Variable

```
COPY_METHOD=sftp
```

setzen. Alle anderen Variablen und Mechanismen wie z.B. [Entfernen der Passwordeingabe](#) entsprechen denen von SCP / SSH (s.o.).

Spezielle Optionen fürs sftp können Sie in der Umgebungsvariable SFTP_OPTS setzen. Es kann z.B. sein, dass das Kopieren großer Dateien in einen Timeout läuft. Bei CSV bietet es sich an, die Dateien zu komprimieren, sie können also setzen:

```
SFTP_OPTS="-oCompression=yes"
export SFTP_OPTS
```

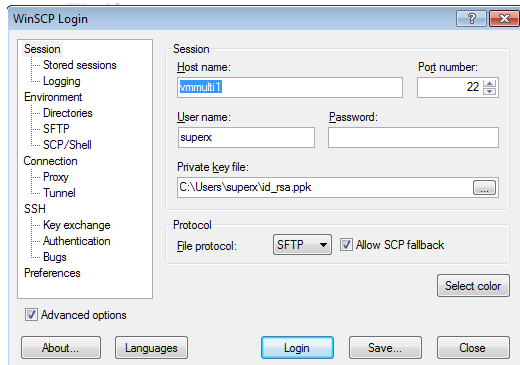
Danach wiederholen Sie das kenn_copy.x

Dies ist nur ein Beispiel. Weitere Optionen für SFTP entnehmen Sie der manpage.

Das Kopieren mit sftp klappt übrigens auch unter Windows z.B. mit dem Programm [winscp](#). Dies wird im folgenden erläutert.

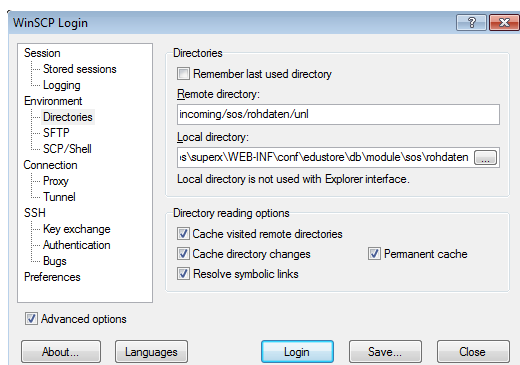
Einrichtung von SFTP unter Windows

Unter Windows ist das Standardprogramm für SFTP die Anwendung [winscp](#). Sie können die Passwordeingabe analog zu Putty [entfernen](#). Danach können Sie eine Session erzeugen:



Im Feld "Private key file" geben Sie den private key ein, den Sie [oben](#) erzeugt haben. Klicken Sie dann auf "Save session", und geben Sie einen kurzen, sprechenden Namen ohne Leerzeichen oder Umlaute, z.B. "superx@vmmulti1"

Sie müssen nun die Verzeichnisse lokal und remote in der Winscp-Session speichern, im Menüpunkt "Directories":



Vergessen Sie auch hiernach nicht, die Session zu speichern.

Danach testen Sie einmal den Zugang. Wenn das klappt können Sie Winscp auch für die Kommandozeile automatisieren. Sie erzeugen zunächst eine Textdatei mit den Kommandos für Winscp, dies enthält auch die zu übertragenden Dateien. Der Aufbau der Datei ist wie folgt:

```
open -Sessionname-
# Upload the file to current working directory
put -Datei-
# Disconnect
close
# Exit WinSCP
exit
```

Die zu übertragenden Dateien sind in der Musterdatei des jew. Moduls in rohdaten/-Modulname-_sftp.txt aufgelistet, Sie müssen lediglich

- den "/" durch einen "\" ersetzen
- das Entladeprotokoll -Modulname-_unload.err und die Datumsdatei superx.datum hinzufügen

Hier ein Beispielscript fürs das Studierenden-Modul:

```
open superx@vmmulti1
# Change remote directory
# Upload the file to current working directory
put sos_unload.err
put superx.datum
put unl\konstanten.unl
put unl\sos_studenten.unl
put unl\sos_hzb.unl
put unl\sos_faecher.unl
put unl\sos_pord_to_stg.unl
put unl\sos_pruefungenext.unl
put unl\sos_pruefungen.unl
put unl\sos_lab_astat_attributes.unl
put unl\sos_stud_loe.unl
put unl\sos_faecher_ext.unl
```

```

put unl\sos_faecher_kontrolle.unl
put unl\sos_pruefungen_kontrolle.unl
put unl\sos_anschri.unl
put unl\sos_parstg.unl
put unl\sos_hsnr.unl
put unl\semester.unl
put unl\cif.unl
put unl\cifx.unl
put unl\k_pvers.unl
put unl\k_stg.unl
put unl\k_stgext.unl
put unl\k_abstgv.unl
put unl\sos_pord.unl
put unl\sos_dipl.unl
put unl\sos_minder.unl
put unl\sos_stud_d.unl
put unl\sos_pords.unl
put unl\sos_porg.unl
put unl\sos_labzuord.unl
put unl\sos_pnrzuord.unl
put unl\sos_gewichtungsvariante.unl
put unl\sos_gewichtungsregel_filter.unl
put unl\sos_gewichtungsregel.unl
put unl\personattribute.unl
put unl\personattributetype.unl
put unl\personattribute_value_list.unl
put unl\stu_update_prot.unl
put unl\hexa_update_prot.unl
put unl\sos_accredited_ects.unl
put unl\sos_pord_orgeinheit.unl

# Disconnect

close

# Exit WinSCP

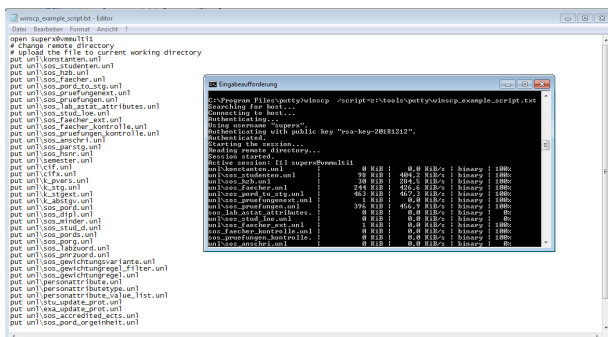
exit

```

Die Datei hat z.B. den Namen winscp_example_script.txt. Dann können Sie das Script aufrufen mit dem DOS-Kommando:

```
winscp /script= Pfad zu \winscp_example_script.txt
```

Hier ein Screenshot wie das dann aussieht:



Das DOS-Kommando lässt sich dann über die Windows-Taskverwaltung automatisieren.

Java-Client zum Entladen von Quell-Datenbanken

Zum Entladen aus dem operativen Vrsystem wird unter Informix dbaccess genutzt. Unter Postgres wird generell der SuperX-JAVA-Client zum Entladen genutzt, denn SuperX benötigt ein spezielles, an Informix angepasstes **CSV-Format**, das sich mit Bordmitteln von Postgres (copy-Befehl) nicht erzeugen lässt. Es kann aber auch sinnvoll sein, aus der Informix-Datenbank mit SuperX-JAVA-Client zu entladen, z.B. wenn Sie kein UNIX-dbaccess auf dem Vrsystem installiert haben.

Wenn Sie das jew. operative VORSYSTEM im PUSH-Verfahren entladen wollen, d.h. die Rohdaten werden auf dem VORSYSTEM entladen und auf den SuperX-Rechner kopiert, dann müssen Sie spezielle Vorkehrungen treffen. SuperX nutzt generell zum Entladen eigene Java-Klassen. Beim Entladen im PULL-Verfahren sind diese Klassen vorhanden, denn die Entladeroutine läuft auf dem SuperX Rechner. Wenn Sie aber PUSH nutzen wollen, werden die SuperX-Java-Klassen auf dem Liefersystem benötigt, und die Entladeroutine muss konfiguriert sein. Im Folgenden nutzen wir das Beispiel "Entladen im Push-Verfahren aus SVA-GX unter Postgres". Gehen Sie dazu wie folgt vor:

- Kopieren Sie die Dateien
 - superx.jar
 - postgresql-*.jar
 - jfor-0.7.2rc1.jar
- vom SuperX-Rechner im Verzeichnis `$$SUPERX_DIR/tomcat/webapps/superx/WEB-INF/lib` auf den Quellrechner in ein Unterverzeichnis lib unter rohdaten (z.B. `/home/sva/superx/rohdaten/lib`). In rohdaten liegt die bisherige Entladeroutine (z.B. `sva_unload.x`).
- Kopieren Sie die `*_ENV.sam`-Datei nach `*_ENV`, also hier `SVA_ENV.sam` nach `SVA_ENV`
- Fügen Sie dann folgenden Passus in die Umgebungs-Datei der Entladeroutine, hier also `SVA_ENV`:

```
#Pfad zu den SuperX-Java-Libraries
#Der JDBC_CLASSPATH enthält alles, was der jdbc-Client in superx für den Datenbankzugriff braucht.
#for *.jar, postgresql*.jar
JDBC_CLASSPATH=$SVA_LOAD_PFAD/lib/superx5.0.jar:$SVA_LOAD_PFAD/lib/postgresql-42.2.19.jar:$SVA_LOAD_PFAD/lib/jfor-0.7.2rc1.jar"
export JDBC_CLASSPATH
```

Wenn dann noch die Variablen `DB_PROPERTIES` und `LOGGER_PROPERTIES` korrekt gesetzt sind, kann die Entladeroutine bei `SX_CLIENT=jdbc` (Wenn Sie unter Windows entladen, oder Informix ohne dbaccess entladen wollen) oder `SX_CLIENT=psql` (wenn Sie Postgres unter UNIX nutzen) mit Java entladen.

Danach können Sie das Entladescript ausführen:

```
sva_unload.x
```

Die Logdatei lautet `sva_unload.err`

Zum Kopieren der Rohdaten zum SuperX-Server passen Sie folgende Variablen an:

Variable	Erläuterung
	Wenn die Rohdaten beim Push-Verfahren nach dem Entladen vom SVA-Rechner auf den BI-Rechner kopiert werden sollen, dann werden für das Script <code>sva_copy.x</code> folgende Umgebungsvariablen benötigt:
<code>COPY_METHOD</code>	Programm, das die Dateien kopiert; <code>rsync</code> , <code>sftp</code> und <code>scp</code> sind wählbar diese müssen dann betriebssystemseitig installiert sein .
<code>REMOTE_DIR</code>	Verzeichnis, in das die Rohdaten auf dem BI-Rechner kopiert werden sollen, in der Regel ist dies <code>/home/superx/db/module/sva/rohdaten</code>
<code>REMOTE_USER</code>	Der Unix-Username auf dem BI-Rechner, in der Regel <code>"tomcat"</code> oder <code>"superx"</code> .
<code>REMOTE_HOST</code>	Der Rechnername bzw. die IP-Nr. des BI-Rechners.

Danach starten Sie

```
sva_copy.x
```

Die Logdatei lautet `sva_copy.err`

Weitere Entladeparameter werden in den modulspezifischen Administrationshandbüchern beschrieben.



Bei neuen SuperX-Versionen reicht es in der Regel, die modulspez. Dateien auszutauschen, also hier `sva_unload.*` etc.

Update eines Moduls: Allgemeines Vorgehen

Wenn das Entladen aus dem VORSYSTEM geklappt hat (sofern es ein VORSYSTEM gibt), können Sie die Daten laden. Zum Update bzw. zum Laden der Rohdaten gehen in das Verzeichnis `$$SUPERX_DIR/db/module/-Modulname-` und führen das Script aus:

```
-Modulname-_update.x
```

Die Logdatei lautet `L_-Modulname-_UPDATE.log`, im Mandantenfähigen Betrieb `"L_-Modulname-_UPDATE-MANDANTID-.log"`.

Für die Aufnahme der Laderoutine in die crontab gibt es im gleichen Verzeichnis Musterscripte nach dem Namensmuster:

```
-Modulname-_update_cron.x
```

Je nach Push/Pull-Szenario können Sie auch den Unload darin starten oder nicht.

Modulupdate in mandantenfähigen Installationen

Der Modulupdate in mandantenfähigen Installation findet in einer SuperX-Installation statt, allerdings werden die einzelnen Scripte mit unterschiedlichen Umgebungsvariablen, wie sie in `SQL_ENV.-MANDANTID-` definiert ist, z.B. `SQL_ENV.FHRO`.

In der `SQL_ENV.-MANDANTID-` werden unterschiedliche Pfade für den jeweiligen `*_LOAD_PFAD` gesetzt, wobei in der Regel die Mandantid ein Unterverzeichnis vom "normalen" `LOAD_PFAD` ist. So ist z.B. beim COB-Modul folgender Pfad anzusetzen:

Normale SuperX-Installation:

```
COB_LOAD_PFAD=$$SUPERX_DIR/db/module/cob/rohdaten
```

Mandantenfähige SuperX-Installation:

```
COB_LOAD_PFAD=$$SUPERX_DIR/db/module/cob/rohdaten/FHRO
```

Unterhalb von `FHRO` befindet sich noch einmal die Entladeroutine sowie das Unterverzeichnis `unl` mit den Rohdaten. Dieses Verzeichnis `FHRO` kann der Einfachheit halber auch ein symbolischer Link auf den gemounteten COB-Rechner sein.

Durch Setzen der Mandantennummer in der Umgebungsvariable `MANDANTID` in der jeweiligen `SQL_ENV` des Mandanten werden die ETL-Scripte anders ausgeführt: Die Logdateien werden jeweils mit der Mandantennummer versehen (z.B. `L_cob_updateFHRO.log`), damit die Übersicht nicht verloren geht und der gleichzeitige Update mehrerer Mandanten in eine rsuperX-Installation möglich ist.

Außerdem können weitreichende Steuerungsmechanismen im Modulupdate eingesetzt werden: Nach jedem ETL-Schritt können optional mandantenspezifische Scripte aufgerufen werden. Diese müssen folgende Namenskonvention einhalten:

```
-Scriptname-_MANDANTID-.sql
```


Also für eine hochschulspezifische Transformation im COB-Modul des Mandanten FHRO wird eine Datei namens

cob_trans_FHRO.sql

mit entsprechenden SQL-Anweisungen angelegt.

Format der Unload Dateien CSV

Generell gilt das Prinzip, daß Daten vom Vorsystem in CSV entladen werden, und dann in sog. Ladedateien hochgeladen werden. Da CSV je nach DBMS unterschiedlich implementiert wird, hier eine kurze Beschreibung des Formats, das im wesentlichen den Vorgaben von Informix LOAD entspricht:

- Zeichenformat: UNIX LATIN1 oder UTF-8
- Feldtrenner: ^
- Satztrenner: Feldtrenner + UNIX NEWLINE
- Zeilenschaltung: Umbrüche innerhalb von Textfeldern sind als \ + NEWLINE codiert
- Feldtrenner, die im Textfeld vorkommen, werden mit "\"" maskiert. Ebenso das Zeichen "\"" selbst
- Die Datumsformate sind bei Datumsfeldern immer im deutschen Format (DD.MM.YYYY) vorgesehen
- Boolean-Werte werden durch "true" oder "false" codiert
- Der Dezimaltrenner ist ".", kein 1000-er Punkt bei Zahlen.
- Leerstrings und Leerzeichen werden als " " (Leerzeichen) exportiert.

Upgrade eines Moduls: Allgemeines Vorgehen

Zum Upgrade bzw. zum Zurücksetzen des Moduls auf den Auslieferungszustand entpacken Sie das Paket in \$SUPERX_DIR und gehen in das Verzeichnis \$SUPERX_DIR/db/module/-Modulname-/upgrade und führen das Script aus:

```
-Modulname-_upgrade.x
```

Ausnahme: beim Kernmodul gibt es i.d.R. ein spezielles Upgrade Script.

Die Logdatei lautet upgrade.log, im Mandantenfähigen Betrieb "upgrade-MANDANTID-.log".

Wenn Sie einen separaten Tomcat-Rechner betreiben, müssen Sie das Paket dort ebenfalls entpacken, und vom Datenbankserver die Datei \$SUPERX_DIR/webserver/tomcat/webapps/superx/WEB-INF/dbforms-config.xml an die gleiche Stelle auf den Tomcat Rechner kopieren. Ein nochmaliges Ausführen des Upgrade Scriptes ist nicht nötig, weil dies nur die Datenbank betrifft.

Hochschulspezifische Anpassung eines Moduls

Dateibasierte hochschulspezifische Anpassung

Nach der Installation bzw. beim Upgrade können Sie hochschuleigene SQL Scripte ausführen lassen. Erzeugen Sie dazu eine Datei

```
$SUPERX_DIR/db/module/-Modulname-/conf/customize.sql
```

bei Mandantenbetrieb

```
$SUPERX_DIR/db/module/-Modulname-/conf/customize-MANDANTID-.sql
```

und füllen Sie diese mit einem beliebigen Inhalt. Das Script wird beim Upgrade des Moduls automatisch am Ende ausgeführt.

Datenbankbasierte hochschulspezifische Laderegeln

Eine Laderoutine gliedert sich in die Schritte

1. Unload: Entladen der CSV Dateien aus dem jew. Vorsystem
2. Load: Upload der CSV Dateien
3. Trans: Transformation der Daten
4. Aggr: Aggregation der Daten
5. Test: Test der Daten
6. System: Standdatum bei Erfolg aktualisieren

Es ist möglich über sog. "Preparation"- oder "Finalize"-[Scripte](#) die Laderoutine am Ende von LOAD bzw. TRANS anzupassen. Dieses Verfahren ist weiterhin möglich, ist aber bzgl. Clustering und Schreibzugriff recht unflexibel.

Es ist auch möglich im Browser hochschulspezifische Scripte am Ende des jeweiligen Schrittes einer Laderoutine anzulegen. Dies funktioniert so: Die Laderoutine führt zunächst den allgemeinen, von uns vorgegebenen Ladeschritt aus. Wenn dann eine Repository-Variable mit einem gewissen Namensschema existiert, wird sie ausgeführt. Das Namensschema lautet

```
<<Kürzel der
Komponente>>_<<Ladeschritt>>
```

- Bei Hauptladeroutinen ist dies
 1. Laden: **Kürzel der Komponente**_load_CUSTOM, z. B. sos_load_CUSTOM
 2. Transformation: **Kürzel der Komponente**_trans_CUSTOM, z. B. sos_trans_CUSTOM
 3. Aggregation: **Kürzel der Komponente**_aggr_CUSTOM, z. B. sos_aggr_CUSTOM
- Bei Unterladeroutinen ist dies **Kürzel der Komponente_ID des Ladeschritts**_CUSTOM, also z. B. cob_trans_fin_busa_CUSTOM für die Unterladeroutine "Lade Buchungen aus FIN"

Um eine Variable neu einzurichten:

- Gehen Sie in den BI-Standardberichten auf Administration/Hochschul-Repository.
- Wählen Sie das jeweilige Sachgebiet und schicken Sie das Formular ab.
- Klicken Sie bei einer Beispielzeile auf **Bearbeiten**.
- Im neuen Fenster klicken Sie oben ganz rechts auf das Symbol **Datensatz kopieren**.
- Definieren Sie die Variable z. B.
 - als Variablenname die ID z. B.: fin_load_CUSTOM
 - Art der Variable: ein für Sie sprechender Schlüssel mit Modulkürzel, z. B. "fin_Laderegeln"
 - als Inhalt der Variable die gewünschten Befehle, z. B.

```
update fin_buch_neu set
ch110_institut=buchungsab
```

- aktiv=1
- gültig seit 1.1.1900
- gültig bis 1.1.3000

- Klicken Sie oben rechts auf das Häkchen-Symbol für **Datensatz einfügen**.
- Schließen Sie das Fenster.

Analog können Sie auch die [customize](#)-Regeln* ins Repository verlagern. Hier heißt die Variable z.B. fin_install_CUSTOM.

Entfernen eines Moduls

Wenn Sie ein Modul nicht mehr benötigen, starten Sie das Script

```
$SUPERX_DIR/db/module/-/Modulname/-/Modulname-_modul_entfernen.x.
```

Dieses Script löscht alle Tabellen, Prozeduren und Abfragen aus der Datenbank, und löscht auch die Einträge im Themenbaum. Danach können Sie den Pfad \$SUPERX_DIR/db/module/-/Modulname-löschen.

Wenn Sie nur die Inhalte der Daten- und Hilfstabellen des Moduls löschen wollen (z.B. aus Datenschutzgründen), ohne das ganze Modul zu deinstallieren, können Sie dies mit folgendem Befehl tun:

```
DOSQL $SUPERX_DIR/db/module/-/Modulname/-/Modulname-_purge_pg.sql (für Postgres)
```

bzw.

```
DOSQL $SUPERX_DIR/db/module/-/Modulname/-/Modulname-_purge_ids.sql (für Informix)
```