

enaio®



Softwaredokumentation

enaio® communicator

Version 11.0

Sämtliche Softwareprodukte sowie alle Zusatzprogramme und Funktionen sind eingetragene und/oder in Gebrauch befindliche Marken der OPTIMAL SYSTEMS GmbH, Berlin oder einer ihrer Gesellschaften. Sie dürfen nur mit gültigem Lizenzvertrag benutzt werden. Die Software sowie die jeweils zugehörige Dokumentation sind nach deutschem und internationalem Recht urheberrechtlich geschützt. Das illegale Kopieren und Vertreiben der Software stellt Diebstahl geistigen Eigentums dar und wird strafrechtlich verfolgt. Alle Rechte vorbehalten, einschließlich der Wiedergabe, Übermittlung, Übersetzung sowie Speicherung mit/auf Medien aller Art. Für vorkonfigurierte Testszenarien oder Demo-Präsentationen gilt: Alle Firmennamen und Personen, die in Beispielen (Screenshots) erscheinen, sind frei erfunden. Eventuelle Ähnlichkeiten mit tatsächlich existierenden Firmen und Personen sind zufällig und unbeabsichtigt.

Copyright 1992 – 2024 by

OPTIMAL SYSTEMS GmbH
Cicerostraße 26
D-10709 Berlin

12.11.2024
Version 11.0

Inhalt

Einleitung	6
Architektur	6
Die Kommunikationskomponenten	6
Konnektoren (Connectors)	7
Transformer	7
Publish-Subscribe	7
Installation	8
enaio® communicator-Monitor	8
enaio® communicator-PWDCrypter	9
Konfiguration	10
Komponenten der Konfiguration	10
Konventionen	10
Abbildung von Kommunikationsstrecken in der Konfiguration	11
Erstellung der Konfiguration	12
Import-Mapping – import-mapping.xml	16
Logging - OscLog	17
Dokumentation	17
enaio® communicator Console	18
Konfiguration	18
Report-Status	18
Dokumentation	20
Monitoring	20
Testen von Nachrichten	21
Dienstbeschreibungen	21
Konfiguration anzeigen	21
Vorhandene Kommunikationskomponenten	22
Dateikonnektor - OscConFile	22
Einleitung	22
Installation	22
Lesen von Dateien aus einem Verzeichnis (inbound)	22
Schreiben von Dateien in ein Ausgabeverzeichnis (outbound)	22
Konfiguration	22
Parameterbereich "outbound"	25
TCP/IP-Konnektoren	26
OscConSocket	26
OscCnSoapEntry	30
Import – OscConImpE	31
Einleitung	31
Voraussetzungen	31
Installation	32
Lizenzen	32
Konfiguration	32
Aufbau von Importdatensätzen	34
Steueranweisungen	36
Dokumenten-Import	38
Beispiele	38
Zusatz	42
Import – OscCnImport	42
Einleitung	42
Voraussetzungen	42
Installation	43
Konfiguration	43
Aufbau von Importdatensätzen	44

Laborimport - oxvbcnli	44
Einleitung	44
Voraussetzungen	45
Installation	45
Konfiguration	45
Ablauf des Laborimports	46
Aufbau der Importdatensätze	46
OxVbCnMrg	47
Einleitung	47
Voraussetzungen	47
Installation	48
Konfiguration	48
Beispiele	49
Bemerkungen	52
OscReadFile	52
Einleitung	52
Voraussetzungen	52
Installation	53
Konfiguration	53
Anwendungsbeispiel	53
OscCnNull	54
Einleitung	54
Voraussetzungen	54
Installation	54
Konfiguration	55
OscCnMSMQ	55
Voraussetzungen	55
Installation	56
Konfiguration	56
OscCnOsEMail	58
Voraussetzungen	58
Installation	58
Konfiguration	58
HL7XML-Transformer – OscHl7XmlCTrans/OscHl7XmlTrans	60
Einleitung	60
Installation	60
Konfiguration	60
Unterschied OscHl7XmlCTrans und OscHl7XmlTrans	60
Beispiel	66
XSLT - OscXsltTransService	72
Einleitung	72
Installation	72
Konfiguration	72
Beispiele	73
OscPassThrough	93
Einleitung	93
Installation	93
Konfiguration	93
OscTrHL7Router	93
Einleitung	93
Voraussetzungen	93
Installation	94
Konfiguration	94
Path zum Messagetyp der HL7-Nachricht	95
Encoding-Transformer – OscTrEncoding	95
Einleitung	95
Konfiguration	95
CSV-Transformer - OscCsvXml	96
Einleitung	96
Voraussetzungen	97
Installation	97
Konfiguration	97
Ausgangsformat von OscCsvXml	98
Beispiele für Konfigurationen und ihre Transformationsergebnisse	100

Fixed-Length-Transformer - OscFixXml	103
Einleitung	103
Voraussetzungen	103
Installation	103
Konfiguration	103
Ausgangsformat von OscFixXml	104
Beispiele für Konfigurationen und ihre Transformationsergebnisse	106
LDT/BDT-Transformer - OscLDTML	107
Einleitung	107
Voraussetzungen	107
Installation	107
Konfiguration	107
Ein- und Ausgangsformat	108
OscTrFallbackrouter	111
Einleitung	111
Voraussetzungen	111
Installation	111
Konfiguration	111
OscTrLookUp	112
Einleitung	112
Voraussetzungen	112
Installation	112
Konfiguration	113
Aufbau von Anfragedatensätze	114
Aufbau von Ergebnisdatensätze	115
OscTrBase64	117
Einleitung	117
Voraussetzungen	117
Installation	117
Konfiguration	117
Beschreibung der Parameter	118
OxTrMuse	120
Einleitung	120
Voraussetzungen	120
Installation	120
Konfiguration	120
OscTrLabMsgLookUp	120
Einleitung	120
Voraussetzungen	121
Installation	121
Konfiguration	121
OscTrDebugPassthrough	122
Einleitung	122
Voraussetzungen	122
Installation	122
Konfiguration	122

Einleitung

Der enaio® communicator ist ein Framework für asynchrone nachrichtenbasierte Kommunikation und wird als Integrationsplattform für die Systemintegration eingesetzt. Über den enaio® communicator können externe Systeme an enaio® angebunden werden.

Über den enaio® communicator können regelbasierte Importe, Exporte, Transformationen von Daten usw. realisiert werden. Dafür können Komponenten des enaio® communicator auf Datenbanken, Dateien, Services usw. zugreifen.

Architektur

Der enaio® communicator besteht aus dem Kernel (enaio® communicator), Komponenten für das Messaging, Logging und die Konfiguration- und Kommunikationskomponenten. Für die Abbildung von Geschäftslogiken gibt es zwei Klassen von Kommunikationskomponenten, die Konnektoren und Transformer.

Der enaio® communicator ist als Windows-Dienst implementiert. Für das Starten und Stoppen des enaio® communicator kann der Dienste-Manager von Windows oder der enaio® communicator-Monitor verwendet werden.

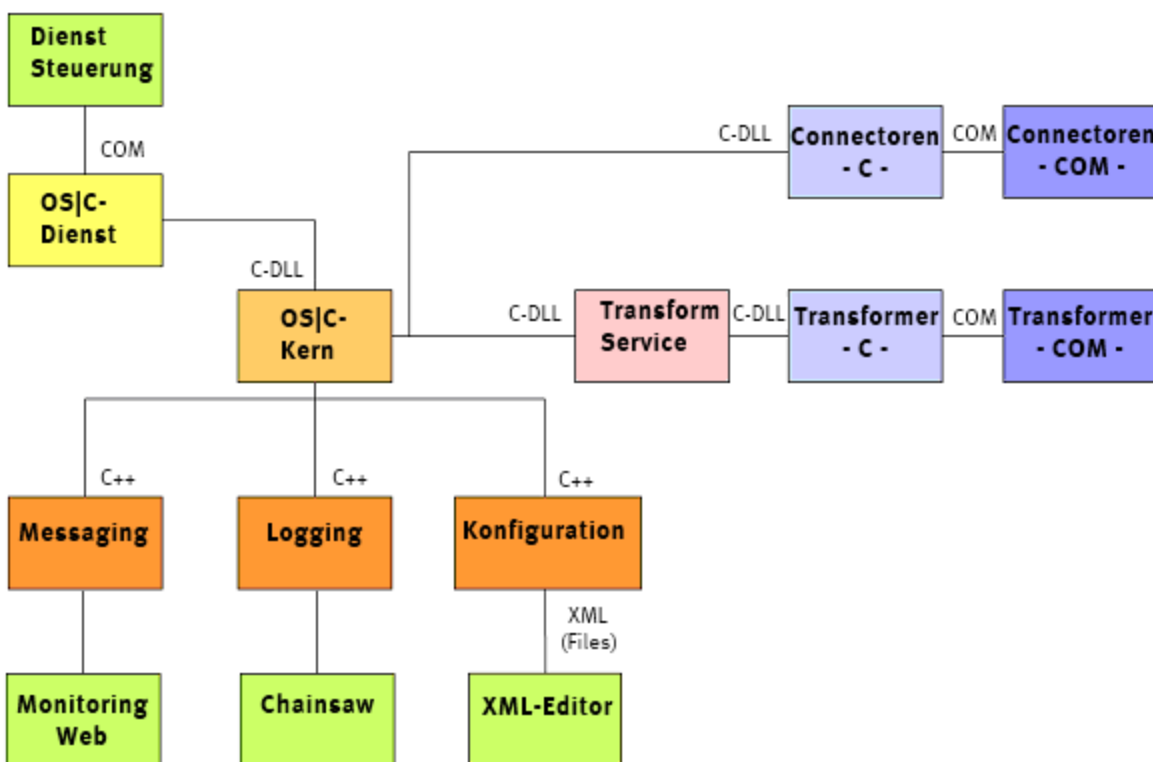


Abbildung 1

Die Kommunikationskomponenten

Von den Kommunikationskomponenten gibt es zwei Klassen, die Transformatoren und die Konnektoren. Diese Komponenten implementieren Geschäftslogiken, die benötigt werden, um Kommunikationstrecken (z.B. HL7-Import) zu realisieren.

Konnektoren (Connectors)

Konnektoren verbinden sich mit externen Systemen:

- Periodisch (receive -> publish), d. h. die Komponente holt Daten und publiziert diese an das interne Messaging-Modul.
- Ereignisgesteuert (subscribe -> send), d. h. die Komponente bekommt Daten (hat diesen Topic also abonniert) und übergibt diese Daten dem externen System, z.B. Daten in die Archivdatenbank importieren

Mit jeweils unterschiedlichen Konnektoren kann auf Daten z.B. per TCP/IP, aus dem Dateisystem, aus Datenbanken, per SOAP zugegriffen werden.

Konnektoren sind jeweils am Anfang und Ende einer Kommunikationsstrecke.

Konnektoren bieten Funktionen zur Steuerung des Verbindungszustandes (connection state management).

Transformer

Transformer implementieren spezielle Transformationsregeln. Beispielsweise kann ein Transformer HL7-Daten oder CSV-Daten (Komma separiert) in ein XML-Format transformieren. Ein Transformer kann mehrere Topics (vgl. Nachrichtentypen) abonnieren bzw. publizieren (vgl. Publish-Subscribe).

Transformer können auf unterschiedliche Weise implementiert werden.

- C-DLL
- XSLT
- COM (beispielsweise in Visual Basic)

Transformer sollen möglichst nicht auf Systemfunktionen zugreifen. Für regelbasierte Validierungen ist es jedoch teilweise notwendig Datenbankabfragen an die Archivdatenbank zu stellen.

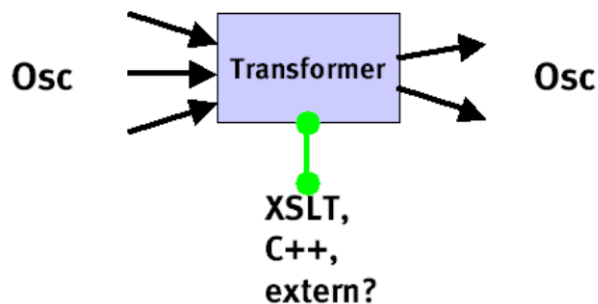


Abbildung 2

Publish-Subscribe

Publish-Subscribe Systeme dienen zur effektiven Verteilung von Information/Nachrichten in verschiedensten Anwendungen. Typischerweise wird eine große Anzahl von Nachrichten (messages) über einen oder mehrere Server (enaio® communicator) an interessierte Nutzer (Konnektoren bzw. Transformatoren) verteilt. Aus der großen Zahl publizierter Nachrichten können die Clients durch ein Abonnement (Subscription) festlegen, an welcher Art von Information sie Interesse haben.

Dieses Prinzip hat den Vorteil, dass sich Absender und Empfänger gegenseitig nicht kennen und somit technisch voneinander unabhängig sind (loose coupling).

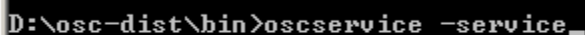
Installation

enaio® communicator wird über das MSI-Paket aus dem Verzeichnis `\Backend\Communicator\` installiert. Mit der Installation werden alle Kommunikationskomponenten ebenfalls installiert. Standardmäßig wird der enaio® communicator unterhalb der Installationsverzeichnisse in das Verzeichnis 'communicator' installiert. In dem Verzeichnis 'bin' werden die Binär-Komponenten des enaio® communicator installiert.

Der enaio® communicator muss nach der Installation aktiviert werden. Dafür muss die Datei `OscComDefs.dll` mit `regsvr32` registriert werden. Diese Aktion registriert die Definitionen der intern verwendeten COM-Interfaces.

Der enaio® communicator muss anschließend ebenfalls registriert werden. Entweder wird der enaio® communicator als Windows-Dienst (`-service`, Abbildung 4) oder COM(`-regserver`) registriert.

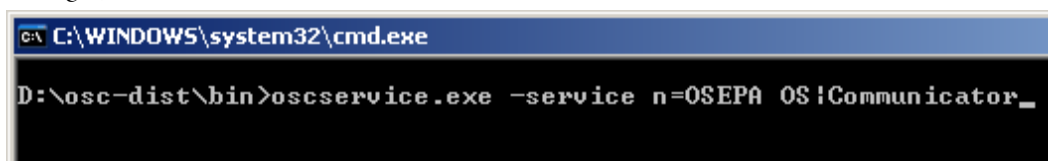
Die Deregistrierung erfolgt mit dem Parameter `-unregserver`. Falls der enaio® communicator als Windows-Service installiert wurde, wird dieser Dienst entfernt.



```
D:\osc-dist\bin>oscservice -service_
```

Abbildung 3

Ab 4.50 SPIV kann der Anzeige-Name für den Dienst dynamisch bei der Installation als Service gesetzt werden (Abbildung 5).



```
C:\WINDOWS\system32\cmd.exe
D:\osc-dist\bin>oscservice.exe -service n=OSEPA OS!Communicator_
```

Abbildung 4

enaio® communicator-Monitor

Der enaio® communicator-Monitor (`OscMonitor.Exe`) ist ein GUI-Programm (grafisch, vgl. Abbildung 5) mit dessen Hilfe der enaio® communicator-Monitor gestartet werden kann. Dieses Programm wird im bin-Verzeichnis des enaio® communicators installiert.

Ist der enaio® communicator als Windows-Dienst registriert worden, kann auch der Dienste-Manager von Windows zum Starten und Stoppen von enaio® communicator verwendet werden. Von der Kommandozeile kann der enaio® communicator über das Kommando `oscservice -autostart` gestartet werden.

Ist der enaio® communicator als Windows-Service registriert, ist der Service-Control-Manager des Betriebssystems für das Starten, Beenden und Neustarten zu verwenden. Wird der Monitor verwendet, ist nicht sichergestellt, dass nach dem sofortigen Wiederstart nach dem Beenden, der enaio® communicator wirklich startet.

Es wird empfohlen den enaio® communicator nur im Testbetrieb nicht als Windows-Service zu betreiben. Im produktiven Betrieb soll der enaio® communicator als Windows-Service installiert werden.

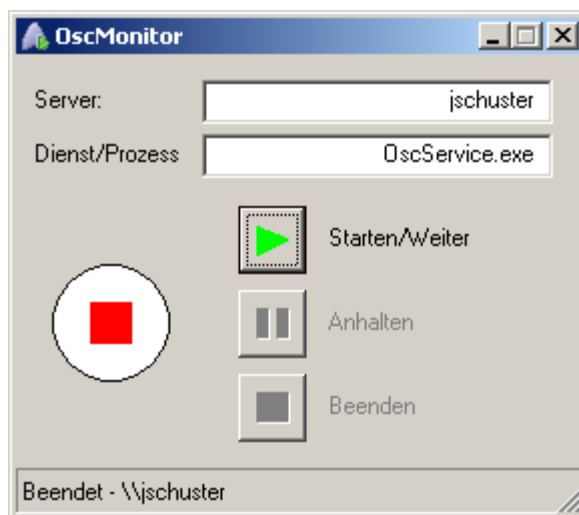


Abbildung 5

enaio® communicator-PWDCrypter

Das Programm enaio® communicator-PwdCrypter (`oscpwdcrypter.exe`) wird im bin-Verzeichnis des enaio® communicators installiert.

Dieses Programm wird benötigt um Passwörter verschlüsselt in den Modulkonfigurationen (vgl. Modulkonfiguration) zu hinterlegen. Komponenten die sich mit einem externen System (Datenbank etc.) verbinden müssen, benötigen Passwortangaben (vgl. Dokumentationen).

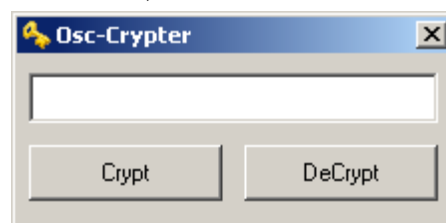


Abbildung 6

Konfiguration

Die Konfiguration des enaio® communicator besteht aus mehreren Dateien, die bis auf die Konfiguration für das Logging XML-Dateien sind. Die Angaben in diesen Konfigurationsdateien steuern den Datenfluss und in welcher Form die Daten bearbeitet werden sollen.

Konfiguriert werden Kommunikationsstrecken (vgl. Abbildung 7), d.h. in der Regel übergibt ein Fremdsystem Daten in einem vereinbarten Format (z.B. HL7, vgl. HL7-Spezifikation). Diese werden aufbereitet (transformiert) und anschließend in die Archivdatenbank importiert.

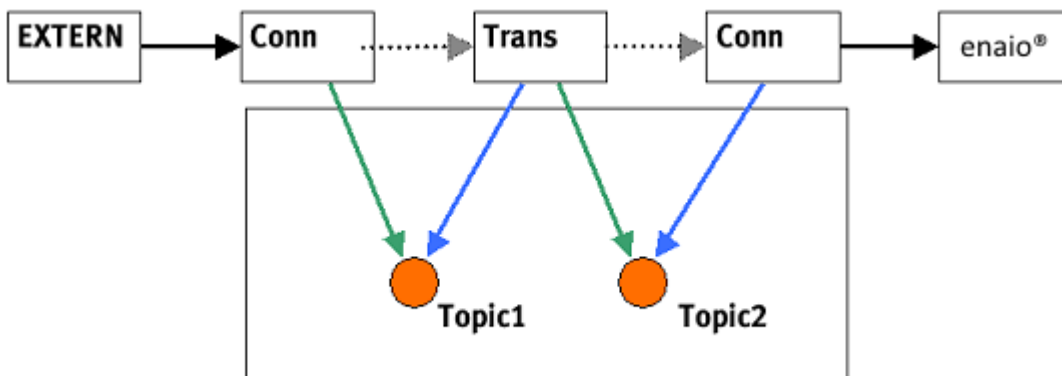


Abbildung 7

Komponenten der Konfiguration

Folgende Komponenten werden konfiguriert:

- Konnektoren (Conn)
- Transformatoren (Trans)

An dieser Stelle wird auf die Dokumentationen der verschiedenen Konnektoren und Transformatoren verwiesen.

Die Komponenten kommunizieren untereinander über Nachrichten, wobei diese Kommunikation nicht direkt erfolgt, sondern über den enaio® communicator geregelt werden.

Jeder Nachrichtentyp ist einem „Topic“ zugeordnet (messagetypes) beschrieben. Die einzelnen Komponenten können Nachrichten publizieren (publish) und abonnieren (subscribe), dabei wird jeweils der gewünschte „Topic“ (messagetype) angegeben, entweder als „input“ für die Komponente oder als ihr „output“.

Konventionen

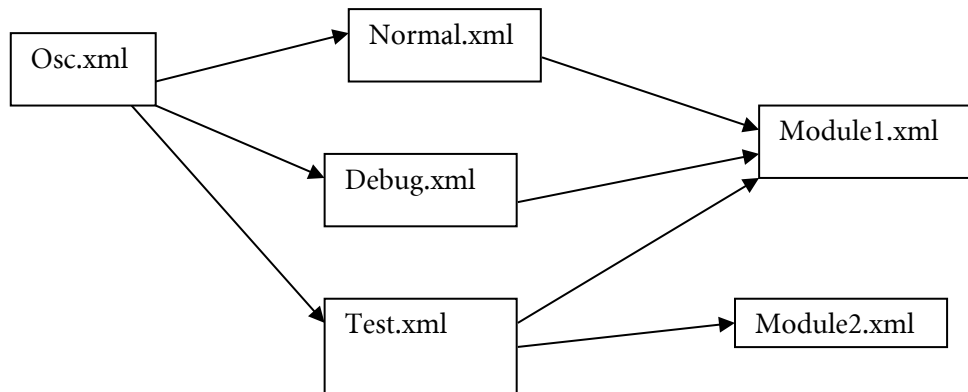
Nachfolgend werden einige Konventionen vorgeschlagen, die die Standardisierung und Lesbarkeit von enaio® communicator-Konfigurationen erhöhen sollen.

Datei-Struktur

enaio® communicator-Konfigurationen haben folgenden Aufbau. Es gibt eine Einstiegskonfiguration (diese hat immer den Namen Osc.cfg, ab Version 5.20 kann auch der Name Osc.xml verwendet werden). Das ist zwingend notwendig. In dieser wird auf die zu verwendene Konfiguration verwiesen, die in einer eigenen Datei hinterlegt wird. Für jede Konfiguration wird eine eigene Datei angelegt. Dadurch kann schnell zwischen Konfigurationen hin und her geschaltet werden. Beispielsweise kann dann eine DEBUG-Version hinterlegt werden, die Nachrichten auf der Festplatte speichert. Diese Konfiguration kann dann bei einer notwendigen Fehlersuche aktiviert werden.

Zusätzliche Konfigurationseinstellungen zu den Konnektoren und Transformatoren (vgl. Dokumentationen) werden in eigenen Modul-Dateien hinterlegt. Hier muss abgeschätzt werden, wann es sinnvoll ist, mehrere Moduldateien zu verwenden, um die Lesbarkeit zu erhöhen.

Beispiel:



Im hier gezeigten Beispiel gibt es drei unterschiedliche Konfigurationen: Eine für den normalen Betrieb, eine zum Debuggen und eine für den Test neuer Komponenten. Alle Konfigurationen verwenden die gleiche Moduldefinition (Module1.xml), in der z.B. IP-Adressen, Verzeichnisse, DB-Connectstrings o.ä. hinterlegt sind. Die Test-Konfiguration benutzt zusätzlich eine weitere Module2.xml, in der zusätzliche Daten für die zu testenden Komponenten hinterlegt sind, um die funktionierende „Normalkonfiguration“ nicht zu verändern.

Namensgebung

Folgende Prefixe sollen verwendet werden.

- `cn` für Konnektoren
- `tr` für Transformatoren
- `r1` für Regeln (vgl. rules)
- `mt` für Nachrichtentypen

Der erste Buchstabe nach dem Prefix wird großgeschrieben. Die gewählten Namen für die Komponenten sollen sprechend sein. Hier einige Beispiele:

Für Konnektoren:

- `cnHL7FileIn`, ein File-Konnektor der HL7-Dateien liest
- `cnHL7SockIn`, ein Socket-Konnektor der HL7-Dateien liest
- `cnImport`, `OscConImpE` Import in die Archivdatenbank

Für Transformation:

- `trHL7ToXml`, `OscHL7XmlCTrans` transformiert HL7-Daten nach HL7-XML
- `trCsvToXml`, `OscCsvXml` transformiert CSV-Daten nach XML

Abbildung von Kommunikationsstrecken in der Konfiguration

Bevor eine Kommunikationsstrecke konfiguriert werden kann, muss bestimmt werden, welche Komponenten (Konnektoren und Transformatoren) beteiligt sind und welche Nachrichten diese verarbeiten sollen. Es muss ebenfalls bekannt sein in welcher Form die Daten übergeben (als Datei oder über TCP/IP) werden.

Als Beispiel wird angenommen, dass HL7-Nachrichten in Dateiform übergeben werden und in die Archivdatenbank importiert werden sollen. Dafür wird eine Komponente benötigt, die die HL7-Nachricht liest, der File-Konnektor. Es wird eine Komponente für den Archivdatenbank-Import

benötigt, OscConImpE. Da keine HL7-Daten, sondern ein spezielles Import-XML-Format (vgl. Dokumentation) verarbeitet wird, müssen die HL7-Daten entsprechend aufbereitet werden. Dafür werden die Komponente OscHL7XmlCTrans OscXsltTransservice verwendet.

Skizze eines HL7-Imports (einfacher Import ohne Validierung) in die Archivdatenbank:

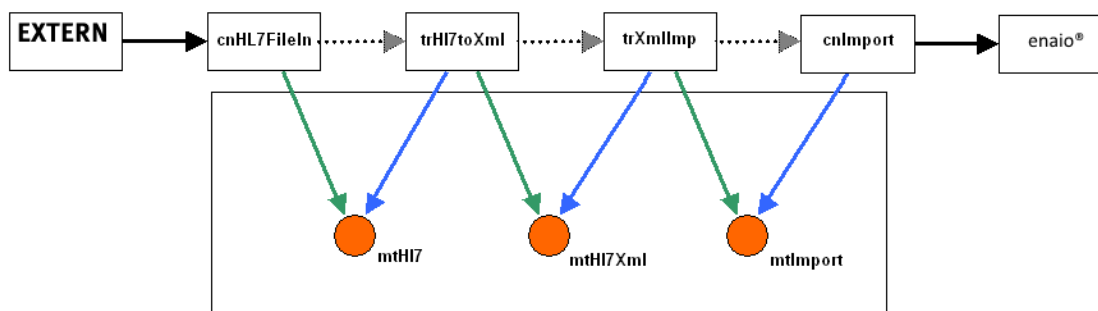


Abbildung 8

Erstellung der Konfiguration

Mit diesen Informationen kann nun die enaio® communicator-Konfiguration erstellt werden.

Folgende Dateien müssen erstellt werden.

- Erstellung der Datei Osc.xml, Einstiegskonfiguration, ermöglicht das Aktivieren und Deaktivieren von Konfigurationen
- Erstellung der Datei Osc.HL7Import.xml, die Konfiguration für die Kommunikationsstrecke.
- Erstellung der Datei Osc.Modules.xml für zusätzliche Konfigurationseinstellungen
- Erstellung des Stylesheets import-mapping.xsl für das Transformieren des HL7-XML zum Import-Format
- Erstellung der Datei OscLog.xml, Konfiguration für das Logging

Weitere komponentenspezifische Konfigurationsdateien wie z.B. die Datei mit der hinterlegten HL7-Grammatik (vgl. Dokumentation OscHL7Trans) werden ebenfalls in dieses Verzeichnis kopiert.

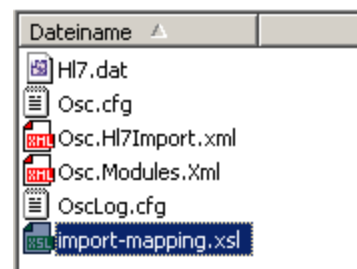


Abbildung 9

Einstieg - Osc.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<osc>
  <configurations>
    <cfg name="Einfacher HL7-Import" file="Osc.HL7Import.xml" active="true"/>
  </configurations>
</osc>
```

Das Monitoring des enaio® communicator wird standardmäßig unterstützt. Dafür werden Konfigurationseinträge (Erläuterung im Abschnitt Monitoring des Status) in der Datei osc.xml benötigt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<osc>
  <configurations>
    <cfg name="Einfacher HL7-Import" file="Osc.HL7Import.xml" active="true"/>
  </configurations>
</osc>
```


Die Konfiguration für diese Kommunikationsstrecke würde folgenden Aufbau haben.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<osc>
  <global>
    <message type="mtHI7" schema="mtHI7.xsd"/>
    <message type="mtHI7Xml" schema="mtHI7Xml.xsd"/>
    <message type="mtImport" schema="mtImport.xsd"/>
    <rule name="rHI7ToXml"
      transformer="C"
      definition="OscHI7XmlCTrans"
      initfile="Osc.Modules.xml">
      <input message type="mtHI7"/>
      <output message type="mtHI7Xml"/>
    </rule>
    <transformation name="trHI7ToXml" rule="rHI7ToXml">
      <input message type="mtHI7" supplier="cnHI7FileIn"/>
      <output message type="mtHI7Xml"/>
    </transformation>
    <rule name="rXmlImp"
      transformer="XSLT"
      definition="import-mapping.xsl">
      <input message type="mtHI7Xml"/>
      <output message type="mtImport"/>
    </rule>
    <transformation name="trXmlImp" rule="rXmlImp">
      <input message type="mtHI7Xml" supplier="trHI7ToXml"/>
      <output message type="mtImport"/>
    </transformation>
    <connector name="cnHI7FileIn"
      definition="OscConFile"
      configuration="Osc.Modules.xml">
      <output message type="mtHI7"/>
    </connector>
    <connector name="cnImport"
      definition="OscComConnector"
      configuration="Osc.Modules.xml">
      <input message type="mtImport" supplier="trXmlImp"/>
    </connector>
  </global>
</osc>
```

Die Nachrichtentypen (messagetypes)

```
<message type="mtHI7" schema="mtHI7.xsd"/>
<message type="mtHI7Xml" schema="mtHI7Xml.xsd"/>
<message type="mtImport" schema="mtImport.xsd"/>
```

Wie in Abbildung 6 wurden entsprechend in der Konfiguration die messagetypes angelegt. Die verwiesenen Schemen in dem Attribut schema müssen nicht physisch vorhanden sein. (Die hier hinterlegte Strukturinformation wird vom enaio® communicator-Kernel nicht verwendet, sie kann aber optional von den Komponenten benutzt werden. Diese Einträge sind vorgesehen für Module, die damit die Struktur der verarbeiteten Nachrichten validieren können, etwa bei XML-Strukturen anhand eines Schemas.)

Die Konnektoren (cnHI7FileIn und cnImport)

Konnektoren werden mit folgenden Attributen konfiguriert.

- name, Name der Komponente (vgl. Konventionen)
- definition, hier wird angegeben welche Art von Komponente verwendet wird. Folgende Angaben sind möglich
OscConFile, Filekonnektor
OscComConnector, COM-Komponente
OscConSocket
- configuration, Moduldatei in der zusätzliche Konfigurationsangaben hinterlegt werden. (vgl. Dokumentationen der Komponenten)

Innerhalb des Konnektors wird ein output und/oder input konfiguriert. Ein output (Inbound-Konnektor) bedeutet, der Konnektor selbst hat einen Output (z.B. der Filekonnektor liest eine Datei und gibt den Inhalt an eine andere Komponente weiter. Ein input bedeutet, der Konnektor bekommt von einer anderen Komponente Daten.

Inboundkonnektor, holt periodisch Daten aus dem Dateisystem.

```
<connector name="cnHI7FileIn"
           definition="OscConFile"
           configuration="Osc.Modules.xml">
  <output message="mtHI7"/>
</connector>
```

Outboundkonnektor, erhält ereignisgesteuert Daten von dem Transformer, der die Importdaten erzeugt (trXmllmp).

```
<connector name="cnImport"
           definition="OscComConnector"
           configuration="Osc.Modules.xml">
  <input message="mtlImport" supplier="trXmllmp"/>
</connector>
```

Die Transformatoren (trHI7ToXml und trXmllmp)

Für jeden Transformer muss zuerst eine Regel(rule) angelegt werden. In dieser wird bekannt gegeben welchen Nachrichten von dem Transformer angenommen und ausgegeben (vgl. Publish-Subscribe) werden.

Regeln werden mit folgenden Attributen konfiguriert.

- name, Name der Komponente (vgl. Konventionen)
- transformer, welche von Art Komponente soll verwendet werden
C
XSLT
Passthrough (vgl. Dokumentation OscPassThroughTransformer)
- definition, welche Komponente verwendet werden soll. Wurde für das Attribut transformer XSLT verwendet, wird hier das Stylesheet angeben. Wurde C angegeben wird hier explizit die C-Komponente (vgl. dazu Dokumentation der vorhandenen Komponenten) angegeben.
OscComTransformer, wird für alle COM-Komponenten verwendet. Es werden dann explizit Konfiguration-Angaben in der Moduldatei erwartet.
OscHI7xmlCTrans, HL7 nach XML transformieren
- initfile, Moduldatei in der zusätzliche Konfigurationsangaben hinterlegt werden. (vgl. Dokumentationen der Komponenten)

Beispiel rlXmllmp:

```
<rule name="rlXmllmp"
      transformer="XSLT"
      definition="import-mapping">
  <input message="mtHI7Xml"/>
  <output message="mtlImport"/>
</rule>
```


Logging - OscLog

Ab Version 5.20 wird mit log4cxx ein log4j-konformes Logging-Framework unterstützt. Daher werden nur noch Logkonfigurationen im log4j-Stil unterstützt.

Es werden der log4j Property-Stil (Name der Konfiguration OscLog.cfg) und das log4j XML-Format (OscLog.xml) unterstützt. Wird das XML-Format verwendet, darf in diesen Konfigurationen nicht auf Datatype-Definitionen verwiesen werden. Für weitere Informationen entnehmen Sie bitte den Dokumentationen von <http://logging.apache.org/>.

Als Log-Viewer kann das Programm „chainsaw“ des gleichnamigen Projektes (<https://logging.apache.org/chainsaw/2.x/>) verwendet werden.

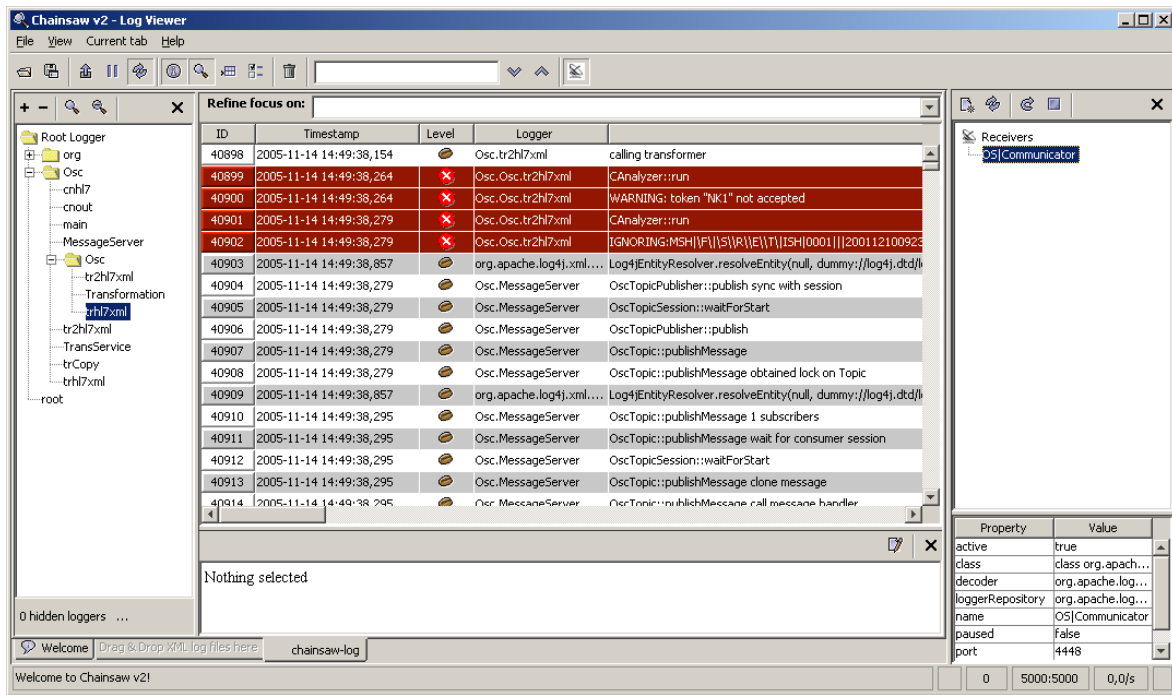


Abbildung 11

Dokumentation

Mit Hilfe des Skriptes “OscMakeDoc.js“, das im doc-Verzeichnis des enaio® communicator abgelegt ist, kann ein HTML-Dokumentation der aktuellen Konfiguration erstellt werden. Diese kann ab Version enaio® communicator über den OscControlWebservice via Webbrowser eingesehen werden.

Für jeden Kommunikationskanal, also pro Inbound-Konnektor wird auch eine Grafik im SVG-Format (scalable vector graphic) erstellt. Diese können sich mit einem SVG-Plugin (für IE von ADOBE) angesehen werden und für die Dokumentation von Projekten verwendet werden.

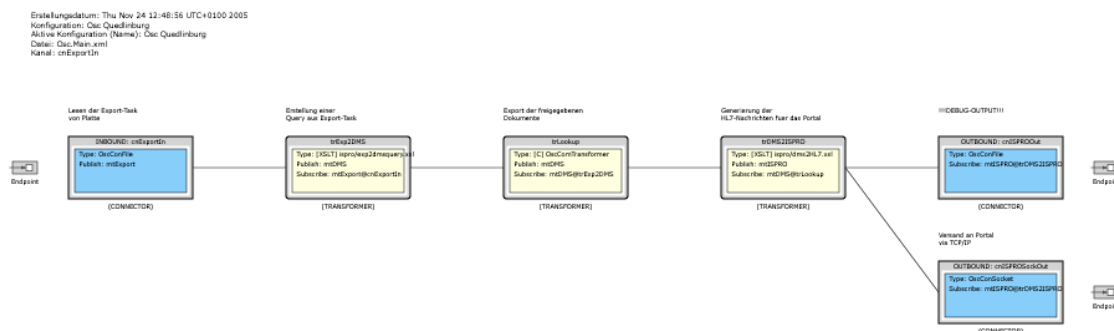


Abbildung 12

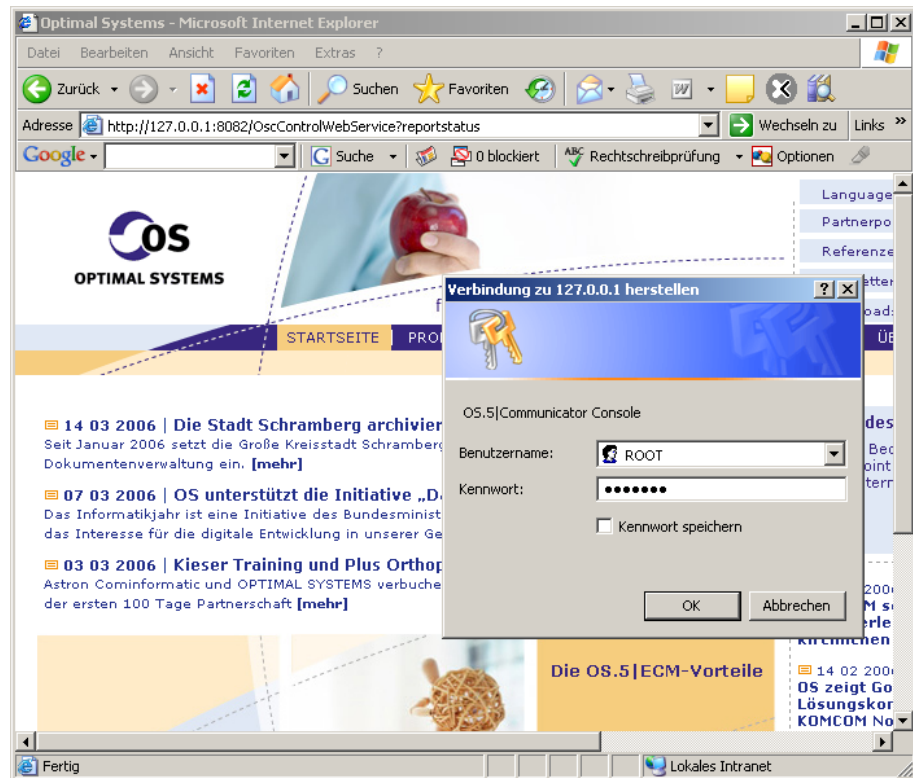


Abbildung 13

Nach dem Aufruf wird der Benutzer aufgefordert den Benutzernamen und das Passwort einzugeben.

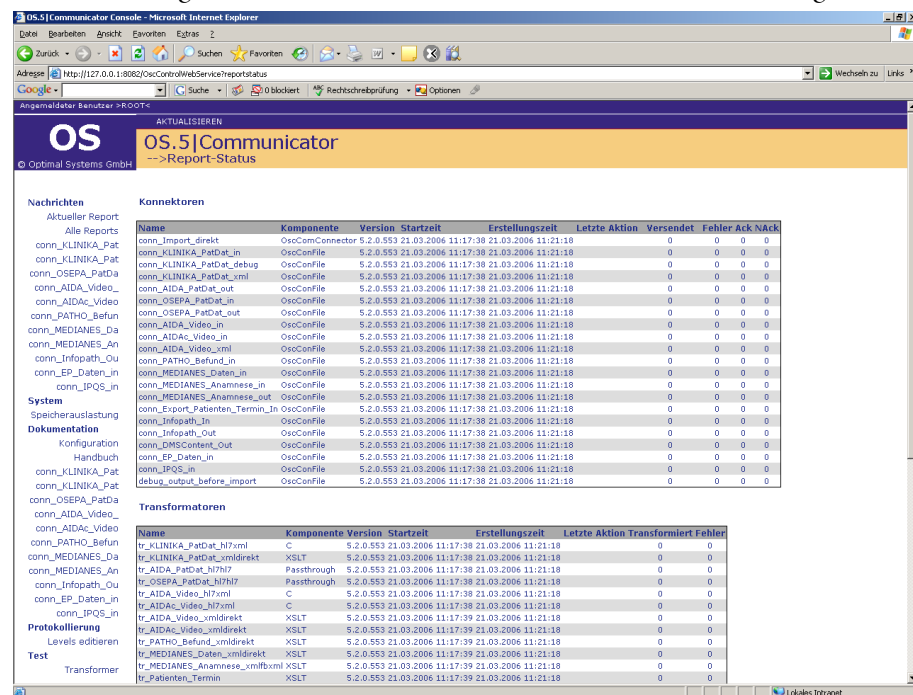


Abbildung 14

Anschließend wird ein aktueller Report-Status im Web-Browser angezeigt. Es wird jeweils eine Liste in Tabellenform für die Konnektoren und die Tranformatoren angezeigt. In den Listen werden Informationen über Art/Name der Komponente und die Anzahl der erfolgreich/fehlerhaft verarbeiteten Nachrichten angezeigt. Über die angezeigten Zeitstempel Start, Erstellungszeit und Letzte Aktion, kann eingesehen werden, kann die Aktivität eingesehen werden. Ist ein Konnektor nicht verbunden (z.B. mit der Archivdatenbank) oder die Komponente entladen wird in der Liste eine Warnung (Name wird rot dargestellt) angezeigt. In dem gezeigten Beispiel weist der Report-Status darauf hin, dass der OscComConnector nicht connected ist. Der Grund kann an dieser Stelle nicht

ermittelt werden. Der Administrator hat nun aber die Möglichkeit gezielt nach betreffenden Meldungen in den Log-Dateien recherchieren.

Der Status der letzten Aktion einer Komponente wird auch durch farbliche Darstellung angezeigt, schwarz = Erfolg und rot = Fehler.

Da der enaio® communicator intern ein asynchrones Nachrichtensystem verwendet, ist bei dem Abrufen von Reporten mit **verzögerten** aktuellen Berichten zu rechnen.

Mit dem Link 'Alle Reports anzeigen' oder dem Aufruf '?allreports' hat der Administrator die Möglichkeit alle Reports seit dem ersten Start des enaio® communicator anzuzeigen. Die Datenbankdatei wird unter dem Namen oscreports.db im bin Verzeichnis der Installation abgelegt.

In der Konnektorentabelle werden die Inbound-Konnektoren mit einem Link dargestellt. Unter diesem Link kann eine grafische Ausgabe für diesen Kommunikationskanal angezeigt werden.

Dokumentation

Unter dem Menüpunkt „Konfiguration“ kann der Benutzer die Konfiguration des enaio® communicator als HTML-Dokument einsehen. In diesem Dokument besteht die Möglichkeit per Links durch die Konfiguration zu navigieren.

Unter dem Menüpunkt „Handbuch“ wird das Handbuch des enaio® communicator angezeigt.

Unter dem Menüpunkt kann der Anwender für jeden Nachrichtenkanal eine grafische Übersicht für den Nachrichtenverlauf einsehen.

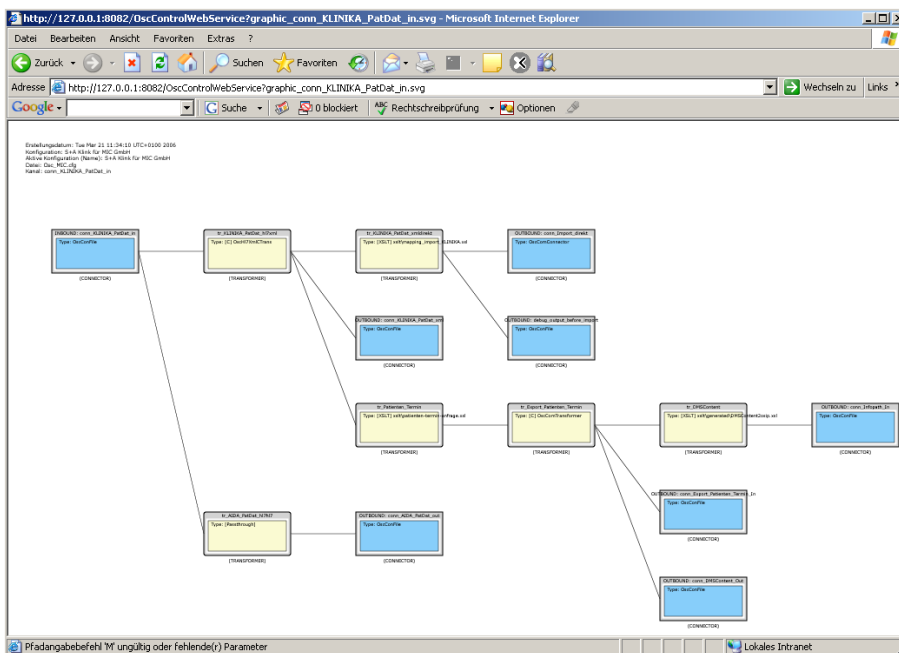


Abbildung 15

Monitoring

Unter dem Menüpunkt „Nachrichten“ kann der Benutzer einsehen, wie viele Nachrichten über einen bestimmten Nachrichtenkanal verarbeitet wurden. Diese Information wird für 24 Stunden gespeichert. So kann z.B. erkannt werden, ob das Nachrichtenaufkommen dem Erwarteten entspricht und gegebenenfalls Gegenmaßnahmen eingeleitet werden.

Unter dem Menüpunkt „System“ wird in der Speicherauslastung der Verlauf des Speicherbedarfs des enaio® communicator für die letzten 24 Stunden angezeigt. So kann erkannt werden, ob es z.B. zu

bestimmten Zeiten Probleme mit der Speicherauslastung gab und diese möglicherweise mit dem Nachrichtenaufkommen im Zusammenhang steht.

Testen von Nachrichten

Unter dem Menüpunkt Test kann der Anwender einzeln jeden Transformer testen. Allerdings muss er Anwender dafür Kenntnis der Konfiguration haben.

Dienstbeschreibungen

Über den OscControlWebSebservice ist es möglich sich die Webservicebeschreibungen von Webservices, die vom enaio® communicator veröffentlicht werden, einzusehen. Dazu fügt der Aufrufer an die URL des OscControlWebservice die gewünschte Beschreibung.

z.B. <http://127.0.0.1:8080/OscControlWebService/OscOnWayMessage.wsdl>.

Konfiguration anzeigen

Wurde eine Dokumentation der Konfiguration erstellt (Siehe Konfiguration|Dokumentation), kann diese über den OscControlWebService via Webbrowser eingesehen werden.

Dazu ist folgender Aufruf notwendig:

<http://127.0.0.1:8080/OscControlWebService/OscCfgDoc.htm>

Wurde der Dateiname der HTML-Konfiguration nach der automatischen Erstellung geändert, muss dieses im Aufruf berücksichtigt werden.

Vorhandene Kommunikationskomponenten

Dateikonnekter - OscConFile

Einleitung

Die Komponente OscConFile ist ein Konnekter zum Lesen und Schreiben von Dateien. Für das Lesen und Schreiben der Dateiinhalte werden temporäre Kopien verwendet, um Konflikte durch gleichzeitigen Zugriff mehrerer Programme auszuschließen.

Installation

Die Komponente OscConFile.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. OscConFile ist Bestandteil jeder enaio® communicator-Installation.

Lesen von Dateien aus einem Verzeichnis (inbound)

Die Eingabe erfolgt zeitgesteuert ("polling"). Die Übernahmedatei wird nach dem Lesen gelöscht. War die enaio® communicator-Aktion insgesamt erfolgreich, wird die Datei in ein "correct"-Verzeichnis verschoben, andernfalls in ein "error"-Verzeichnis. In diesen Verzeichnissen werden datumsabhängig Unterverzeichnisse erzeugt, in denen die Kopien der Übernahmedatei unter ihrem ursprünglichen Namen landen. Ist dort bereits eine gleichnamige Datei vorhanden, so wird diese zuvor umbenannt. Bei der Umbenennung wird ein Dateiname generiert, der den aktuellen Zeitstempel enthält.

Schreiben von Dateien in ein Ausgabeverzeichnis (outbound)

Die Ausgabe erfolgt ereignisgesteuert. Es werden verschiedene Methoden zur Bestimmung des Ausgabedateinamens angeboten (s.u.) Beginnt eine Ausgabe an OscConFile (outbound) mit der Zeile
%OSC-1.0 1 filename=DATEINAME.EXT
so wird automatisch der enthaltene Dateiname DATEINAME.EXT für die Dateiausgabe verwendet. In diesem Falle bleiben alle weiteren Methoden zur Bestimmung des Ausgabe-Dateinamens unberücksichtigt.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

Beispiel für einen inbound-Konnekter.

```
<connector name="cn_in" definition="OscConFile" configuration="modules.xml">  
  <output messagetype="mthl7" standard="true"/>  
</connector>
```

Beispiel für einen outbound-Konnekter.

```
<connector name="cn_out" definition="OscConFile" configuration="modules.xml">  
  <input supplier="trhl7xml" messagetype="mthl7xml"/>  
</connector>
```

Beispiel für einen in-/outbound-Konnektor.

```
<connector name="cn" definition="OscConFile" configuration="modules.xml ">
  <output messagetype="mthl7" standard="true"/>
  <input supplier="trhl7xml" messagetype="mthl7xml"/>
</connector>
```

Außerdem setzt OscConFile einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Hier muss beachtet werden, ob der Konnektor inbound oder outbound ist. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="cn">
  <general timeout="2000"/>
  <inbound>
    <directory>F:</directory>
    <tmp-directory>F:</tmp-directory>
    <filenames>*.nc</filenames>
    <header>filename</header>
    <semaphore>*.sem</semaphore>
    <encoding>OEM</encoding>
    <insert-xml-header>OEM</insert-xml-header>
    <!-- Ab Version 4.50 SPIII -->
    <!-- Wird keine Datendatei zu der Semaphore gefunden, wird die Semaphore automatisch gelöscht.
    Auf true setzen, wenn Semaphore, die zeitlich vor den Datendateien in das Eingangsverzeichnis
    geschrieben werden, gelöscht werden sollen
    Auf false setzen, wenn Semaphore-Dateien nur dann gelöscht werden sollen, wenn die Datendateien
    bereits abgearbeitet wurden.-->
    <auto-delete-semaphore>true</auto-delete-semaphore>
  </inbound>
  <outbound>
    <directory>F:</directory>
    <tmp-directory>F:</tmp-directory>
    <filenames>*.nc</filenames>
    <split-node>/asimport/import</split-node>
    <split-select></split-select>
    <encoding>OEM</encoding>
    <!-- Ab Version OS.5|ECM -->
    <!--Standard aus Kompatibilitätsgründen true -->
    <replace-header>>false</replace-header >
    <!-- Ab Version OS.5|ECM -->
    <!--Standard aus Kompatibilitätsgründen false -->
    <return-false>true</return-false >
  </outbound>
</module>
```

Parameterbereich "general"

Das Attribut **timeout** gibt das Polling-Intervall für die *inbound*-Seite des Connectors in Millisekunden an. Wird keine Eingabe gefunden oder war die letzte Eingabe nicht erfolgreich, so wartet die Komponente für die Dauer **timeout**, bis das nächste Mal die Eingabe geprüft wird. Nach einer erfolgreichen Eingabe wird unabhängig davon sofort erneut versucht, die nächste Eingabe zu lesen. Soll der Connector ausschließlich als Ausgabe-Connector verwendet werden, so sollte dieser Wert auf "0" gesetzt werden. In diesem Fall findet kein Polling statt.

Parameterbereich "inbound"

Diese Parameter gelten für das Lesen von Dateien (Eingabe)

directory

Pflichtparameter. Verzeichnis, in dem nach Dateien gesucht wird.

tmp-directory

In diesem Verzeichnis werden während der Übernahme temporäre Dateien abgelegt. Default: gleiches Verzeichnis wie **directory**. Beim Import wird zunächst eine Kopie der in **directory** gefundenen Eingabedatei in **tmp-directory** erzeugt, aus dieser Datei werden die Daten gelesen. Nach erfolgreichem bzw. fehlerhaftem Abschluß der Übernahme wird die Originaldatei aus **directory** in das Verzeichnis **tmp-directory/correct** bzw. **tmp-directory/error** kopiert und anschließend die temporäre Datei gelöscht.

filenames

Wird für filenames in einem inbound und outbound das Pattern *.* verwendet, muss das Attribut header=filenames verwendet werden.

1. inbound - Angabe des Suchmusters für Eingabedateien.
2. Angabe des Eingabeformats. hier sollten Muster wie abs*.* oder *.hl7 angewendet werden.

header

Das Element muss den Textwert "filename" enthalten. Den gelesenen Daten wird vor der Weiterverarbeitung im enaio® communicator eine Zeile

```
%OSC-1.0 1 filename=DATEINAME.EXT
```

vorangestellt.

Tipp: Erhält eine Ausgabe für OscConFile (outbound) eine solche Kopfzeile, so wird automatisch der enthaltene Dateiname für die Dateiausgabe verwendet. Somit kann der OscConFile-Connector optional zum Verschieben von Dateien unter Beibehaltung des Dateinamens verwendet werden. Standardmäßig wird diese Zeile beim Herausschreiben mit einem Fileconnector wieder entfernt. Soll dieses Verhalten verhindert werden, muss im outbound-Bereich der Parameter replace-header (s.u.) konfiguriert werden.

semaphore

Suchpattern für Semaphore-Dateien. Semaphore-Dateien werden von einigen Systemen zur Synchronisation der Dateiübernahme verwendet. Wird eine Datei gefunden, die auf das Semaphore-Pattern passt, so wird der Dateiname der Übergabedatei bestimmt durch Ersetzen der File-Extension der Sem-Datei durch die File-Extension, die im "filenames"-Pattern enthalten ist. Mit diesem Verfahren wird also sichergestellt, dass auf die Übergabedatei erst dann zugegriffen wird, wenn die zugehörige Sem-Datei vorhanden ist. Die Sem-Datei wird erst nach erfolgter Übernahme der Originaldatei gelöscht.

encoding

Optionale OEM-To-ANSI Zeichensatzkonvertierung bei der Eingabe. Zulässig ist derzeit nur der Wert "OEM"

insert-xml-header

Bei einem eingelesenen XML-Dokument wird direkt unter dem root-Element ein zusätzliches Element mit dem angegebenen Namen (default: osc-filetransfer) erzeugt. Dieses Element enthält als Text den Namen der eingelesenen Datei. Dieser Mechanismus stellt eine für XML-Eingaben einsetzbare Alternative zu der o.b. "Osc-Kopfzeile" dar. (Für XML-Ausgaben wird dieser Mechanismus *nicht* benutzt).

daily-folder

Mit dieser Option kann gesteuert werden, ob täglich error- und correct-Verzeichnisse angelegt werden. Standardmäßig ist der Wert auf 'true' gesetzt. Wird false konfiguriert, werden die Dateien direkt im error/correct-Verzeichnis und nicht in Tagesordnern abgelegt.

zip-backup

Mit dieser Option kann gesteuert werden, dass die Dateien in error- und correct-Verzeichnissen gezippt werden. Konnten alle Dateien erfolgreich gezippt werden, werden diese aus dem Verzeichnis gelöscht. Standardmäßig ist der Wert auf 'false' gesetzt. Diese Option ist nur anwendbar, wenn daily-folder mit 'true' konfiguriert wurde.

Parameterbereich "outbound"

Diese Parameter gelten für das Schreiben von Dateien (Ausgabe)

directory

Zielverzeichnis (Pflichtparameter).

tmp-directory

In diesem Verzeichnis werden während der Ausgabe temporäre Dateien abgelegt. Default: gleiches Verzeichnis wie **directory**. Es wird zunächst eine temporäre Ausgabedatei in **tmp-directory** erzeugt, in diese Datei werden die Daten geschrieben. Anschließend wird diese Datei aus **tmp-directory** in das Zielverzeichnis **directory** verschoben und umbenannt.

filenames

Muster für die Erzeugung des Ausgabe-Dateinamens. Enthält das Element einen C-Formatstring, wird dieser mit einem fortlaufenden Zähler gefüllt. Bei expliziter Angabe eines Ausgabenamens in der Kopfzeile bleibt dieser Wert unberücksichtigt und stattdessen wird der Dateiname aus der Kopfzeile verwendet. Derzeit gibt es die Möglichkeit den Ausgabenname mit einer hochgezählten Dateinummer zu versehen (Beispiel hl7%05u.xml; %05u ist der Platzhalter, der dafür sorgt das diese Zahl mit bis zu Fünf führenden Nullen aufgefüllt wird) oder einen Zeitstempel in den Dateinamen zu integrieren (Beispiel hl7%ts.xml, %ts wird durch einen Zeitstempel im Format JJJMMTTSSMMSS000, Zeitstempel wird auf Millisekunden-Ebene erstellt)

encoding

Optionale ANSI-To-OEM Zeichensatzkonvertierung bei der Ausgabe. Zulässig ist derzeit nur der Wert "OEM"

Bei den beiden folgenden Optionen handelt es sich um Parameter, die speziell für die Generierung von "AS-Import"-Dateien im CSV-Format in Verbindung mit einer vorhergehenden speziell angepassten XSL-Transformation eingeführt wurden. Von einer Verwendung für andere Zwecke wird abgeraten. Perspektivisch werden auch AS-Importe mit einem alternativen, vollständig XML-basierten Verfahren abgebildet.

split-node

Das mit diesem Parameter verbundene Verfahren dient der Erzeugung mehrerer separater ASCII-Dateien aus einem einzigen entsprechend strukturierten XML-Eingabestring (insbesondere für enaio®-Importe).

Dieser Parameter ist nur dann sinnvoll, wenn ein outbound OscConFile-Connector einen passenden XML-String als Eingabe erhält.

Wenn dieser Parameter angegeben ist, erfolgt eine Aufspaltung der Eingabe in mehrere ASCII-Einzeldateien. Die Aufspaltung erfolgt anhand des angegebenen XPath-Ausdruckes (siehe selectNodes) zunächst in ein XML-Nodeset. Der String-Wert der einzelnen NodeListElements wird dann in jeweils

eine eigene Datei geschrieben. Der jeweilige Dateiname wird anhand des Wertes von 'split-select' bestimmt (siehe dort).

Eine Aufspaltung in mehrere XML-Dateien ist an dieser Stelle nicht vorgesehen. Zu diesem Zweck sollte besser eine entsprechende Kaskade von XSL-Transformationen im Sinne von "Filtern" verwendet werden,

split-select

Das Argument ist ein XPath-Ausdruck, der relativ zu dem "split-node" ausgewertet wird. Das Ergebnis der Auswertung von string(split-select) wird bei der Erzeugung des Ausgabe-Dateinamens verwendet, indem es dem aus "filename" berechneten Dateinamen vorangestellt wird. Beispieltransformation

replace-header

Dieses Konfigurationstag ist als Pendant zu header im inbound-Bereich zu verstehen. Standardmäßig wird der Wert mit true belegt, also die Header-Zeile entfernt. Soll dieses Verhalten verhindert werden, muss der Wert false angegeben werden.

daily-folder

Dieses Konfigurationstag wird verwendet, um das Erstellen eines täglichen Verzeichnisses zu erzwingen. Standardmäßig ist die Einstellung false.

TCP/IP-Konnektoren

OscConSocket

Einleitung

Die Komponente OscConSocket ist ein Konnektor zum Versenden und Empfangen von Daten via TCP/IP-Socketverbindungen. Die Komponente kann entweder als TCP/IP-Server oder als TCP/IP-Client konfiguriert werden. In der Regel gibt es zwei verschiedene Modi (vgl. unten), die Komponente zu betreiben. Das Begriffspaar client/server beschreibt die Rollen beim Aufbau der TCP/IP-Verbindung.

TCP/IP-Verbindungen ermöglichen eine bidirektionale Kommunikation. Üblicherweise sendet ein TCP/IP-*Client* zunächst Daten (*sendet* eine „Nachricht“) an den Server und erhält anschließend vom *Server* einen Antwortdatensatz (nach *Empfang* wird eine „Quittung“ übergeben). Diese Arbeitsweise wird durch die folgenden Einstellungen ermöglicht:

- Modus ist „server“ und HL7-Modus ist „receive“
- Modus ist „client“ und HL7-Modus ist „send“

Die Rollen send/receive können aber auch vertauscht werden: Der Client verbindet sich mit dem Server und wartet jeweils bis der *Server* eine Nachricht *sendet*, nach dem *Empfang* beantwortet/quittiert der *Client* die Nachricht und wartet dann erneut.

Da im medizinischen Bereich HL7 (vgl. HL7-Spezifikation) das bevorzugte Austauschformat ist, wurde OscConSocket für die Verarbeitung von HL7-Daten implementiert (HL7-MLLP, minimum lower layer protocol).

Die Erzeugung/Verarbeitung der nach HL7-Protokoll erforderlichen ACK/NAK-Nachrichten (acknowledge bzw. not acknowledge, Bestätigungsmeldungen des empfangenden System) erfolgt selbständig intern im Konnektor. Dabei wird die Message-Control-ID der Nachrichten verwendet.

Diese Komponente kann darüber hinaus auch den Inhalt empfangener ACK- und NAK-Nachrichten an andere Komponenten weiterleiten. Zum Beispiel kann der externe Empfänger bestätigen, dass er eine Nachricht vom enaio® communicator erhalten hat und mitteilen ob er die Daten verarbeiten konnte oder nicht. Mit Hilfe dieser Nachricht kann der enaio® communicator auf den Versand der Daten entsprechend reagieren, beispielsweise eine Fehlermeldung generieren und in die Archivdatenbank importieren (vgl. OscConImpE).

Installation

Die Komponente OscConSocket.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. OscConSocket ist Bestandteil jeder enaio® communicator-Installation.

Server-Betrieb

Im Serverbetrieb wird durch OscConSocket in einem eigenen Thread ein TCP-Listener erstellt, der alle eingehenden Anforderungen an den konfigurierten Anschluss (hostname und port) behandelt.

Anschließend wartet der Listener auf eingehende Verbindungen.

Im Receiver-Modus (das ist die Regel) werden die Daten vom Socket gelesen und an nachfolgende Komponenten (Transformer bzw. Konnektoren) weiter gereicht.

Im Sende-Modus werden dem verbundenen Client Nachrichten gesendet.

Nach der Verarbeitung sendet OscConSocket entweder eine ACK (bestätigt) oder NACK-Message (nicht bestätigt) an das System, dass die Nachricht gesendet hat.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

```
<connector name="cn_Sock_In"
  definition="OscConSocket"
  configuration="modules.xml ">
  <output messagetype="mt_versand"/>
</connector>
```

Außerdem setzt OscConSocket einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus.

```
<module name="cn_Versand_In">
  <mode>server</mode>
  <hl7-mode>receive</hl7-mode>
  <hostname>127.0.0.1</hostname>
  <port>4111</port>
  <wait-before-connect>3</wait-before-connect>
  <wait-no-response>40</wait-no-response>
  <sending-application>OS4X</sending-application>
  <sending-facility>Osc_OSEPA_1MED</sending-facility>
  <receiving-application>PROMPT</receiving-application>
  <receiving-facility>OS</receiving-facility>
  <processing-id>P</processing-id>
  <version-id>2.4</version-id>
  <start-of-block>0x0b</start-of-block>
  <end-of-data>0x1c</end-of-data>
  <end-of-block>0x0d</end-of-block>
</module>
```

mode

Im Server-Modus muss hier „server“ eingetragen werden.

hl7-mode

In der Regel wird hier „receive“ angegeben. Es kann auch „send“ angegeben werden (vgl. oben).

hostname

Die IP bzw. der Netzwerkname des Rechners für den der Listener erstellt werden soll.

port

Portnummer auf der gelauscht werden soll.

wait-before-connect

Mit diesem Attribut kann eine Verzögerung vor der Verbindungserstellung eingestellt werden.

wait-no-response

Maximale Zeitspanne für eine Antwort der Gegenstelle.

sending-application

Sendene Applikation (vgl. HL7-Spezifikation, MSH-3)

sending-facility

Sendende Einrichtung. Zusätzliche Beschreibung zur sendenden Applikation. (vgl. HL7-Spezifikation, MSH-4)

receiving-application

Empfangene Applikation (vgl. HL7-Spezifikation, MSH-5)

receiving-facility

Empfangene Einrichtung. Zusätzliche Beschreibung zur empfangenden Applikation. (vgl. HL7-Spezifikation, MSH-6)

processing-id

Mit dieser ID kann bestimmt werden, in welcher Betriebsart sich die Anwendung befindet (vgl. HL7-Spezifikation, MSH-11). Mögliche Einstellungen sind:

- D - Debugging
- P - Production
- T - Training

version-id

Verwendete HL7-Spezifikation.

start-of-block

Startzeichen der Nachricht. Standard ist 0x0B (Start Block character, vgl. Appendix C Lower LayerProtocols der HL7-Spezifikation).

end-of-data

Endzeichen nach dem Datenteil der Nachricht. Standard ist 0x0D (End Block character, vgl. Appendix C Lower LayerProtocols der HL7-Spezifikation).

end-of-block

Endzeichen der gesamten Nachricht. Standard ist 0x0D (Carriage Return, vgl. Appendix C Lower

LayerProtocols der HL7-Spezifikation).

Client-Betrieb

Im Client-Betrieb verbindet sich OscConSocket mit einem anderen Socket-Server (vgl. Server-Betrieb). Eine Verbindung kann demzufolge nur erstellt werden, wenn ein Socket-Server auf den konfigurierten TCP/IP-Einstellungen lauscht (hostname und port).

Im Sende-Modus (mode=send, das ist die Regel) sendet die Komponente, wenn eine Nachricht für den Socket-Server an die Komponente übergeben wird, die Daten an den Socket-Server.

Im Receiver-Modus (mod=receive) wird die Komponente von dem Server mit Nachrichten versorgt. Der Unterschied in den Modis „send“ und „receive“ besteht darin, dass die Komponente als Sender (send) Daten an den Server sendet

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

```
<connector name="cn_Sock_Out"
           definition="OscConSocket"
           configuration="modules.xml ">
  <input supplier="tr_Hl7" messagetype="mt_ver_hl7"/>
  <!-- Output für ack-nack -->
  <output messagetype="mt_ack"/>
</connector>
```

Außerdem setzt OscConSocket einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus.

```
<module name="cn_Sock_Out">
  <general timeout="0"/>
  <hl7-mode>send</hl7-mode>
  <mode>client</mode>
  <hostname>194.94.160.143</hostname>
  <port>5820</port>
  <!-- Ack-Nack -->
  <forward-external-ack>true</forward-external-ack>
  <start-of-block>0x0b</start-of-block>
  <end-of-data>0x1c</end-of-data>
  <end-of-block>0x0d</end-of-block>
  <wait-before-connect>3</wait-before-connect>
  <wait-no-response>40</wait-no-response>
</module>
```

general timeout

Für einen OutBound-Connector idt dieser Wert explizit auf 0 zu setzen. Damit wird die Receive-Funktion, die für die InBound-Funktionalität benötigt wird abgeschaltet. Der Standardwert ist 1000.

mode

Im Server-Modus muss hier „client“ eingetragen werden.

hl7-mode

In der Regel wird hier „send“ angegeben. Es kann auch „receive“ angegeben werden (vgl. oben).

hostname

Die IP bzw. der Netzwerkname des Rechners mit dem sich verbunden werden soll.

port

Port über den kommuniziert werden soll.

wait-before-connect

Mit diesem Attribut kann eine Verzögerung vor der Verbindungserstellung eingestellt werden.

wait-no-response

Maximale Zeitspanne für eine Antwort der Gegenstelle.

start-of-block

Startzeichen der Nachricht. Standard ist 0x0B (Start Block character, vgl. Appendix C Lower LayerProtocols der HL7-Spezifikation).

end-of-data

Endzeichen nach dem Datenteil der Nachricht. Standard ist 0x0D (End Block character, vgl. Appendix C Lower LayerProtocols der HL7-Spezifikation).

end-of-block

Endzeichen der gesamten Nachricht. Standard ist 0x0D (Carriage Return, vgl. Appendix C Lower LayerProtocols der HL7-Spezifikation).

forward-external-ack

Die ACK- bzw. NACK-Nachricht des Socket-Servers wird verarbeitet. Dadurch ist es möglich nach dem Versand von Daten auf die Reaktion (Bestätigung oder Fehler) des empfangenen System zu reagieren und entsprechende Aktionen einzuleiten.

OscCnSoapEntry

Der Konnektor OscCnSoapEntry dient dem Empfang von Soap-Nachrichten via TCP/IP. Als Schnittstelle dient eine Webservice-Description (OscOneWayMessage.wsdl im bin-Verzeichnis des enaio® communicator). Aus dieser können Clients entsprechende Proxyobjekte generieren und auf sehr schnelle Weise eine Kommunikation zum enaio® communicator aufbauen. Mit Hilfe dieses Konnektors können Clients sehr einfach plattformunabhängig, über Rechnergrenzen hinweg und eine über standardisierte Technik Nachrichten an den enaio® communicator gesendet werden. Die Nachrichten werden in einer internen Nachrichtenschlange gespeichert. Wenn diese Nachrichten persistent zur Verfügung gestellt werden müssen, können dafür z.B. die Komponenten OsconFile oder OscCnMSMQ verwendet werden. Für weitere Fragen wenden Sie sich bitte an den Support von Optimal Systems.

Installation

Die Komponente OscCnSoapEntry.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. OscCnSoapEntry ist Bestandteil jeder enaio® communicator-Installation.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

```
<connector name="cn_Soap_In"
           definition="OscCnSoapEntry"
           configuration="modules.xml ">
  <output messagetype="mt_ack"/>
```

```
</connector>
```

Außerdem setzt OscConSocket einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus.

```
<module name="cn_Soap_In">
  <general timeout="0"/>
  <upload-path> D:\dvp\las500\OscServer\CPP\data\soap\in</upload-path>
  <mode>client</mode>
  <port>5820</port>
</module>
```

port

Port über den kommuniziert werden soll.

upload-path

Diese Konfigurationseinstellung ist für zukünftige Versionen vorgesehen. Damit wird es möglich sein, Images via Soap an den enaio® communicator zu senden.

Erstellung von Clients für OscCnSoapEntry

Für die Erstellung von Proxy-Klassen für den Zugriff auf OscCnSoapEntry wird die WSDL OscOneWayMessage.wsdl zur Verfügung gestellt. Diese Dienstbeschreibung wird im bin-Verzeichnis der enaio® communicator-Installation installiert. Für das Generieren von Proxy-Klassen werden Tools von verschiedenen Anbietern angeboten.

Import – OscConImpE

Einleitung

Die Komponente OscConImpE ist ein Konnektor (vgl. enaio® communicator-Dokumentation), der für den enaio® communicator entwickelt und als ActiveX-Exe realisiert wurde.

Mit OscConImpE.Exe können Importe in die Datenbank des Archiv-Servers über den enaio® communicator realisiert werden. Es können Indexdaten und Dokumente importiert werden. Im Gegensatz zu den automatischen Aktionen, die zeit- bzw. ereignisgesteuert aufgerufen werden, sind Importe über den enaio® communicator nachrichtengesteuert. D.h., wird eine Importnachricht an den enaio® communicator gesendet, verarbeitet dieser diese Nachricht entsprechend seiner Konfiguration und sendet dann die Importdaten an OscConImpE. Die Struktur der Import-Daten muss dem Schema asimport.xsd entsprechen. Über OscconImpE können komplexe Strukturen in die Archivdatenbank importiert werden. Beispielsweise ist es möglich ein Importdokument zu erstellen, dass Daten für mehrere Schränke, Register und Dokumente enthält. Das Import-Schema wird entsprechend der Releaseversion der Komponente gepflegt.

Importe in die Archivdatenbank, die über den enaio® communicator realisiert werden, können durch den Einsatz zusätzlicher Komponenten (in der Regel Transformer, die bestimmte Vor- und Nachbedingungen testen) sehr komplexe Logiken abbilden.

Voraussetzungen

Da die Verbindung zum Archiv-Server über die Client-Komponenten realisiert wurde, müssen die Client-Komponenten ebenfalls auf dem System installiert sein.

OscConImpE benötigt die Lizenzen ASC OSE.

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Axbasics.dll	enaio®	Client-Komponente

Axclnt.dll	enaio®	Client-Komponente
Oxidxdat.dll	enaio®	Admin-Komponente
Osccomdefs.dll	enaio®	enaio® communicator-Komponente
Axvbxml.dll	enaio®	enaio® communicator-Komponente
Msoxml4.dll	Microsoft	
MS ADO	Microsoft	Ab Version 2.1

Für den Import werden Anmeldeinformationen eines Benutzers benötigt, in dessen Kontext der Import ausgeführt werden soll. Werden Konfigurationseinstellungen gewählt, die Zugriffe auf das Dateisystem benötigen (vgl. Konfiguration), ist darauf zu achten, dass der Import-Benutzer die ausreichenden Rechte besitzt.

Installation

Die Komponente OscConImpE wird standardmäßig im admin-Verzeichnis der Installation installiert. Da es sich um eine ActiveX-Exe handelt muss diese registriert werden (mit dem Schalter –regserver).

Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein.

Die Komponente OscConImpE.dll benötigt für eine korrekte Lauffähigkeit folgende im gleichen Verzeichnis registrierte Komponenten:

- Oxidxdat.dll
- Oxmljsc.dll
- Axclnt.dll
- Axbasics.dll

Lizenzen

Diese Komponente benötigt die Lizenzen ASC und OSE.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<connector name="cnimp"
           definition="OscComConnector"
           configuration="modules.xml">
  <input messagetype="mt_osimport" supplier="trimport"/>
</connector>
```

Außerdem setzt OscConImpE einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="cnimp">
  <!--benötigte Lizenzen -->
  <lic>ASC OSE</lic>
  <!--mit oscpwdcrypter.exe verschlüsseltes Passwort -->
  <pwd>00000000000000</pwd>
  <!--Benutzer, in dessen Kontext der Import läuft -->
  <usr>HL7-Import</usr>
```



```

<!-- Angabe in Sekunden (ab OS:4.2) -->
<wait-before-connect>60</wait-before-connect>
<!-- temporäres Verzeichnis (ab OS:4.2) -->
<tmp-directory>D:\osc-dist\data\JiveX\out\preview-temp</tmp-directory>
<!-- sollen Import-Dokumente gespeichert werden (ab OS:4.2) -->
<persist-doc>true</persist-doc>
<!-- temporäres Verzeichnis zum Speichern von Dokumenten (ab OS:4.2) -->
<doc-path>D:\osc-dist\data\JiveX\out\doc-path</doc-path>
<!-- Verzeichnis für Dateien fehlerhafter Importe (ab OS:4.2) -->
<persist-error>D:\osc-dist\data\JiveX\out\Importfehler</persist-error>
<!-- True oder False) -->
<delete-import-documents>True</delete-import-documents>
<!-- Standard ist auf true gestellt -->
<no-task-correct>>false</no-task-correct>
<!-- Dokumente archivierbar setzen, Standard = False -->
<archivable>>false</ archivable >
<!-- Alle Tabellenzeilen vor dem Import löschen, Standard = False -->
<delete-table-rows>True</ delete-table-rows>
<!-- Prog-ID, wird vom enaio® communicator benötigt -->
<ComProgId>OscConImpE.Conn</ComProgId>
</module>

```

Als Mindestkonfiguration werden folgende Angaben benötigt:

lics
pwd
usr
ComProgId

Für die Anmeldeinformationen ist ein Benutzer im System einzurichten, in dessen Kontext der Import läuft, beispielsweise „oscimport“.

lics

Benötigte Lizenzen ASC OSE

pwd

Mit oscpwdcrypter.exe verschlüsseltes Passwort

usr

Benutzer, in dessen Kontext der Import läuft

wait-before-connect

Wird verwendet um eine verzögerte Anmeldung an den Archivserver zu erreichen. Dadurch können Synchronisationsprobleme des enaio® communicators ausgeglichen werden. Als Standard wird 60 Sekunden durch die Komponente gesetzt. So wartet OscConImpE die entsprechende Zeitspanne vor der Anmeldung und beginnt erst anschließend mit dem Import. Verhindert wird damit das Phänomen, dass Importdateien während des Starts des enaio® communicators in das error-Verzeichnis kopiert werden.

tmp-directory

Die Einstellung tmp-directory wird benötigt, wenn zusätzlich zu den Indexdaten von Dokumenten Dokumente (z.B. XML, Word oder JPEG) importiert werden sollen. Diese werden in diesem

Verzeichnis temporär gespeichert und nach dem Import wieder gelöscht.

persist-doc

Sollen die Dokumente permanent vom enaio® communicator im Dateisystem gespeichert werden, muss das Attribut auf „true“ gesetzt werden und der `doc-path` angegeben werden.

doc-path

Pfad, in dem Dokumente permanent gespeichert werden sollen.

delete-import-documents

Steuerung, ob dass importierte Dateidokumente nach dem Import gelöscht werden soll oder nicht. Es wird darauf hingewiesen, dass dieses Attribut nur auf `true` gesetzt werden soll, wenn die zu importierenden Dateien nicht von anderen Systemen benötigt werden und die Berechtigung für das Löschen vorhanden ist.

no-task-correct

Wenn diese Einstellung auf 'false' gestellt wird, führt eine fehlerhafte Import-Task dazu, dass der enaio® communicator die Verarbeitung als fehlerhaft gemeldet (Tipp: Steuerung `correct` und `error-Verzeichnis`) bekommt.

archivable

Dokumenten archivierbar setzen. Defaultwert: `false`

delete-table-rows

Alle Tabellenzeilen vor dem Import löschen. Defaultwert: `false`

ComProgId

Com-Id des Connectors: `OscConImpE.Conn`

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Für die Erstellung des Importdokuments wird meistens eine XSLT-Transformation innerhalb des enaio® communicator realisiert.

Aufbau von Importdatensätzen

Importdatensätze müssen dem Schema `asimport.xsd` (ab Version 4.20.x) entsprechen.

Grundsätzlich ist ein Importdatensatz wie folgt aufgebaut:

```
<asimport>
<task>
  <ord>
    <reg>
      <doc>
        <file/>
      </doc>
    </reg>
  </ord>
</task>
</asimport>
```

Über die Komponente OscConImpE kann auf Tabellen-Controls importiert werden. Dafür ist prinzipiell folgender Datensatzaufbau notwendig. In den Beispielen finden Sie ein ausführliches Beispiel.

```
<asimport>
<task>
  <ord>
    <reg>
      <doc>
        <file/>
        <table>
        </table>
      </doc>
      <table>
      </table>
    </reg>
    <table>
    </table>
  </ord>
</task>
</asimport>
```

Innerhalb des Tags `asimport` können jedoch mehrere `tasks` abgelegt werden. Dadurch können Importe auf mehrere Schrankobjekte innerhalb eines Importdatensatzes realisiert werden. Es ist auch möglich verschiedene Schrankobjekte anzusprechen, also z.B. einen Import auf Patientendaten und auch auf Adressendaten durchzuführen.

Die Anzahl der `ord`-, `reg`- und `doc`-Tags ist beliebig. Ein `doc`-Tag kann auch direkt unterhalb des `ord`-Tags liegen (Dokumente können auch direkt Schrankobjekten zugeordnet werden).

Register können über die Attribute `name` oder `dbname` spezifiziert werden. Entweder wird das Attribut „`name`“ (entspricht dem Namen auf der Client-Maske, z.B. Patient) oder „`dbname`“ (entspricht dem Namen in der Datenbank-Datenbankbezeichner, z.B. register1) verwendet.

Mit Hilfe des `file`-Tags können auch Dokumente zu den Indexdaten importiert werden. Für die Anwendung dieser Option wird hier auf die Beispiele verwiesen.

Zur Veranschaulichung sollen folgende zwei schematischen Beispiele dienen.

Import auf :

Patient/(Schrank), Aufenthalt (Register) und Leistung(Dokument)

Adressen(Schrank)

```
<asimport>
<task>
  <ord name="Patient">
    <reg name="Aufenthalt">
      <doc name="Leistung"/>
    </reg>
  </ord>
  <ord name="adressen">
  </ord>
</task>
</asimport>
```

Patient (Schrank), Aufenthalt (Register) und Leistung(Dokument)

Adressen

```
<asimport>
<task>
  <ord name="Patient">
```

```

        <reg name="Aufenthalt"/>
        <doc name="Leistung"/>
</ord>
    <ord name="adressen">
</ord>
</task>
</asimport>
    
```

Der Unterschied zwischen diesen beiden Importdatensätzen ist, dass beim ersten das Dokument unterhalb des Registerobjekts und beim zweiten direkt im Schrankobjekt abgelegt werden soll.

Die Darstellung der Importdatensätze in diesem Abschnitt ist als rein schematisch anzusehen. Für reale Datensätze wird an dieser Stelle auf die Beispiele dieser technischen Information verwiesen.

In diesem Schema kann eingesehen werden, welche Formate durch OscConImE unterstützt werden.



Asimport.xsd

Steueranweisungen

In den Importdokumenten werden verschiedene Anweisungen verwendet, um den Import zu steuern.

Anweisung	Bedeutung
select	Dient der Auswahl eines Objekts, das geändert werden soll oder innerhalb dessen ein anderes Objekt gefunden werden soll. Das Objekt muss genau einmal gefunden werden.
select-or-insert	Dient der Auswahl eines Objekts, das geändert werden soll oder innerhalb dessen ein anderes Objekt gefunden werden soll. Wird das Objekt nicht gefunden, wird dieses neu angelegt. Das Objekt darf maximal einmal gefunden werden.
insert	Das Objekt wird neu angelegt. Es darf jedoch noch nicht vorhanden sein.
update	Mit dieser Anweisung wird angezeigt, dass das entsprechende Feld geändert werden soll.

Feldattribute

Feldattribute müssen angegeben werden, um Datenfelder aus den Archivtabellen zu selektieren, einzufügen oder zu ändern. Zwei Angaben müssen gemacht werden

- Name des Felds (name bzw. dbname)
- Wert des Felds (value)

Bei der Angabe des Namens gibt es zwei Möglichkeiten. Entweder wird das Attribut „name“ (entspricht dem Namen auf der Client-Maske, z.B. Patientenummer) oder „dbname“ (entspricht dem Namen in der Datenbank-Datenbankbezeichner, z.B. feld1) verwendet. An dieser Stelle wird empfohlen die Namen von der Client-Maske zu verwenden, da diese sprechender sind und dadurch für den Ersteller der Stylesheets für die Erzeugung der Importdatensätze übersichtlicher sind.

Beispiele für die Anwendung der Steueranweisungen und Feldattribute

Die select-Anweisung

Diese Anweisung dient der Selektion eines Objekts. Dieses Objekt kann dann z.B. geändert werden oder nur als Auswahl für ein weiteres Unterobjekt dienen. Die select-Anweisung kann auch in Kombination verwendet werden.

1. Auswahl eines Objekts anhand des ID-Felds der Client-Maske

```
<select name="ID" value="12345678"/>
```

2. Auswahl eines Objekts anhand des Datenbankbezeichners feld1:

```
<select dbname="feld1" value="12345678"/>
```

3. Auswahl eines Objekts anhand eines Kombination aus Feldern aus der Client-Maske:

```
<select name="Nachname" value="Mustermann"/>
```

```
<select name="Vorname" value="Manfred"/>
```

4. Auswahl eines Objekts anhand einer Kombination aus Feldern aus der Client-Maske und Datenbankbezeichnern:

```
<select dbname="Nachname" value="Mustermann"/>
```

```
<select name="feld1" value="125789"/>
```

Die insert-Anweisung

Diese Anweisung dient dem Einfügen von Objekten. Da Objekte eindeutig sein müssen, werden alle insert-Anweisungen dafür verwendet sicherzustellen, dass ein noch kein Objekt mit diesen Kriterien in der Datenbank vorhanden ist.

1. Einfügen eines Objekts mit ID und Name:

```
<insert name="ID" value="12345678"/>
```

```
<insert name="Name" value="Mustermann"/>
```

Diese Kombination darf vor dem Einfügen nicht in der Datenbank gefunden werden.

Es können dieselben Kombinationen wie bei der select-Anweisung angewendet werden.

Die select-or-insert-Anweisung

Diese Anweisung dient dem Selektieren von Objekten. Wird dieses Objekt jedoch nicht gefunden, wird es angelegt. Das Objekt darf also kein- bzw. einmal in der Datenbank vorhanden sein. Diese Anweisung kann auch in Kombination angewendet werden.

1. Einfügen eines Objekts mit ID und Name:

```
<select-or-insert name="ID" value="12345678"/>
```

Es können dieselben Kombinationen wie bei der select-Anweisung angewendet werden.

Die update-Anweisung

Diese Anweisung wird für das Ändern bzw. Einfügen von Datenfeldern eines Objekts verwendet. Die update-Anweisung kann nur in Kombination mit den 3 vorher genannten Anweisungen verwendet werden. Wurde über diese Steueranweisungen ein Objekt selektiert, führt dies zu einem klassischen Update. Wurde ein neues Objekt angelegt, wird die update-Anweisung als Insert verstanden.

1. Update eines Objekts, das zuvor anhand der ID ausgewählt wurde:

```
<select name="ID" value="12345678"/>
```

```
<update name="Name" value="Mustermann"/>
```

Das data-Tag

Das data-Tag ist der container für die Steueranweisungen. Das heißt innerhalb eines Objekt-Tags(ord, reg oder doc) muss ein data-Tag angelegt werden, in denen dann die Steueranweisungen (die u.a. als konkrete Importdaten interpretiert werden) abgelegt werden.

```
<data>
```

```
  <select-or-insert name="Name" value="Mustermann"/>
```

```
  <update name="Vorname" value="Manfred"/>
```

```
  <update name="Geburtsdatum" value="19.12.1960"/>
```

```
</data>
```

Import auf mehrzeilige Textfelder

Der Import von Daten auf mehrzeilige Textfelder wird ab der Version 4.50.208 unterstützt. Da Umbrüche nicht innerhalb von XML-Attributen dargestellt werden können, müssen Werte mit Umbrüchen innerhalb im Textknoten übergeben werden.

```
<update name="Bemerkungen">mehrzeilige Textfelder
```

mehrzeilige Textfelder

mehrzeilige Textfelder

mehrzeilige Textfelder

</update>

Dokumenten-Import

Ab der Version 4.20.x der OscConImpE.Exe ist es möglich zusätzlich zu den Indexdaten eines Dokuments ein echtes Dokument in die Archivdatenbank zu importieren. Beispielsweise können XML-Dokumente, Word-Dateien oder Bilder (JPEG) importiert werden.

Dafür muss der Pfad zu dem Dokument in dem Importdatensatz bekannt gegeben werden. Eine Ausnahme stellt das XML-Dokument dar. Dieses kann auch im Importdatensatz eingebettet werden. Dadurch können flexibel mit Hilfe von Stylesheets XML-Dokumente importiert werden, die im Client mit Hilfe des XML-Moduls angezeigt werden können. Für die konkrete Anwendung wird hier auf die Beispiele verwiesen.

Beispiele

Im Folgenden soll anhand von einigen Beispielen gezeigt werden, welche Importstrukturen durch OscConImpE verarbeitet werden können.

Indexdatenimport

Der erste Beispieldatensatz zeigt eine simple Importdatei für Indexdaten auf Schrank- Register- und Dokumentenebene. Auf Schrankebene wird für das Objekt „Patient“ anhand des Namens entschieden, ob das Objekt angelegt bzw. geändert werden soll (select-or-insert). Auf Registerenebene wird das Objekt „DICOM“ angelegt oder selektiert. Auf Dokumentenebene wird auf das Objekt „Bild“ anhand der Study Instance UID entschieden, ob das Objekt angelegt werden muss oder lediglich geändert werden soll.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<asimport xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc">
  <task>
    <ord name="PATIENT">
      <data>
        <select-or-insert name="Name" value="Mustermann"/>
        <update name="Vorname" value="Manfred"/>
        <update name="Geburtsdatum" value="19.12.1960"/>
      </data>
      <reg name="DICOM">
        <data>
          <select-or-insert name="DICOM" value="DICOM"/>
        </data>
        <doc name="Bild">
          <data>
            <select-or-insert name="Study Instance UID"
              value="123456789"/>
            <update name="Study Date" value="12.02.2004"/>
            <update name="Patient's ID" value="111"/>
            <update name="Patient's Name"
              value="Mustermann"/>
            <update name="Patient's Birthday"
              value="19.12.1960"/>
            <update name="Patient's Sex" value="M"/>
            <update name="Accession Number" value="1"/>
            <update name="Modality" value="5"/>
            <update name="Study Description"
              value="Musterpatient"/>
          </data>
        </doc>
      </reg>
    </ord>
  </task>
</asimport>
```

```

value="Arzt"/>
<update name="Study ID" value="123456789"/>
<update name="Series Number" value="1"/>
<update name="Referring Physician's Name"
value="Arzt"/>
</data>
</doc>
</reg>
</ord>
</task>
</asimport>

```

Der zweite Beispieldatensatz zeigt eine simple Importdatei für Indexdaten auf Schrank- und Dokumentenebene. Auf Schrankebene wird auf das Objekt „Patient“ anhand der Namen, Vorname und Geburtsdatum ein Objekt selektiert. Wird ein Objekt gefunden, wird auf Dokumentenebene anhand der Study Instance UID entschieden, ob neues Objekt „Bild“ angelegt werden muss oder lediglich geändert werden soll. Zusätzlich wird auf Orderebene eine Adresse importiert, wobei das Selektionskriterium die ID des Datensatzes ist.

Selektion des Ordner-Objekts über das Register

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<asimport xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc">
  <task>
    <ord name="PATIENT">
      <data>
        <select name="Name" value="Mustermann"/>
        <select name="Vorname" value="Manfred"/>
        <select name="Geburtsdatum" value="19.12.1960"/>
      </data>
      <doc name="Bild">
        <data>
          <select-or-insert name="Study Instance UID"
value="123456789"/>
          <update name="Study Date" value="12.02.2004"/>
          <update name="Patient's ID" value="111"/>
          <update name="Patient's Name" value="Mustermann"/>
          <update name="Patient's Birthday" value="19.12.1960"/>
          <update name="Patient's Sex" value="M"/>
          <update name="Accession Number" value="1"/>
          <update name="Modality" value="5"/>
          <update name="Study Description" value="Musterpatient"/>
          <update name="Study ID" value="123456789"/>
          <update name="Series Number" value="1"/>
          <update name="Referring Physician's Name" value="Arzt"/>
        </data>
      </doc>
    </ord>
    <ord name="Adressen">
      <data>
        <select-or-insert name="ID" value="1500"/>
        <update name="Name" value="Mustermann"/>
        <update name="Straße" value="Musterstraße 100"/>
        <update name="Postleitzahl" value="15500"/>

```

```

        </data>
    </ord>
</task>
</asimport>

```

Sind keine Daten aus einem Ordner bekannt, kann die Selektion über Registerdaten erfolgen. Der Aufbau des Importdatensatzes ist dann folgend. In diesem Import wird über das Attribut Fallnummer aus dem Register Aufenthalt das zugehörige Objekt aus dem Ordner Patient selektiert. Voraussetzung ist, dass ein eindeutiges Kriterium im Register für die Selektierung des Ordner-Objekts vorhanden ist. Um diese Option zu nutzen ist neben der Lizenz ASC auch die Lizenz OSE notwendig.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<asimport xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc">
  <task>
    <ord name="PATIENT">
      <reg name="Aufenthalt">
        <data>
          <select-or-insert name="Fallnummer" value="1234569"/>
        </data>
        <doc name="Bild">
          <data>
            <select-or-insert name="Study Instance UID"
value="123456789"/>
            <update name="Study Date" value="12.02.2004"/>
            <update name="Patient's ID" value="111"/>
            <update name="Patient's Name"
value="Mustermann"/>
            <update name="Patient's Birthday"
value="19.12.1960"/>
            <update name="Patient's Sex" value="M"/>
            <update name="Accession Number" value="1"/>
            <update name="Modality" value="5"/>
            <update name="Study Description"
value="Musterpatient"/>
            <update name="Study ID" value="123456789"/>
            <update name="Series Number" value="1"/>
            <update name="Referring Physician's Name"
value="Arzt"/>
          </data>
        </doc>
      </reg>
    </ord>
  </task>
</asimport>

```

Dokumentenimport

Um Dokumente (XML, Word, JPEG) zu importieren wird innerhalb des doc-Tags ein file-Tag angelegt. Innerhalb von diesem Tag wird ein insert-Tag erwartet. Für den Import mehrerer Dokumente wird eine Liste von insert-Tags angelegt. Im Attribut „name“ des insert-Tags kann angegeben werden, um welchen Typ es sich handelt. Grundsätzlich werden die gleichen Dokumententypen unterstützt, die über den Client importiert werden können. Ein eingebettetes XML-Dokument wird in ein document-tag innerhalb von insert angelegt. Über die Attribute version und encoding des document-Tags kann die XML-Deklaration beeinflusst werden. Für ein Dokument in

Dateiform muss der Pfad in dem Attribut value des insert-Tags angegeben werden.

Im ersten Beispiel werden in das Importdokument eingebettete XML-Daten als XML-Dokument in die Archivdatenbank importiert. Diese können dann mit Hilfe des XML-Moduls angezeigt werden. Für die Konfiguration kann in dem Abschnitt Konfiguration nachgesehen werden.

```
<doc name="Pathologie-Bericht">
  <data>
    <select-or-insert name="ID" value="123456789"/>
  </data>
  <file>
    <data>
      <insert name="xml">
        <document version="1.0" encoding="ISO-8859-1">
          <pathologiebericht>
            Dies ist ein Pathologiebericht.
          </pathologiebericht>
        </document>
      </insert>
    </data>
  </file>
</doc>
```

Im zweiten Beispiel wird eine Bilddatei aus dem Dateisystem für das Bildobjekt importiert. In dem Attribut file wird der Dateipfad zu der zu importierenden Datei angegeben.

```
<doc name="Bild">
  <data>
    <select-or-insert name="Study Instance UID" value="123456789"/>
    <update name="Study Date" value="12.02.2004"/>
    <update name="Patient's ID" value="111"/>
    <update name="Patient's Name" value="Mustermann"/>
    <update name="Patient's Birthday" value="19.12.1960"/>
    <update name="Patient's Sex" value="M"/>
    <update name="Accession Number" value="1"/>
    <update name="Modality" value="5"/>
    <update name="Study Description" value="Musterpatient"/>
    <update name="Study ID" value="123456789"/>
    <update name="Series Number" value="1"/>
    <update name="Referring Physician's Name" value="Arzt"/>
  </data>
  <file>
    <data>
      <insert name="dia" file="c:\bilder\mustermann.jpeg"/>
    </data>
  </file>
</doc>
```

Tabellenfelder

Über die Komponente OscConImpE kann auf Tabellen-Controls importiert werden. Der Aufbau der Datensätze ist folgender, exemplarisch im Ordner:

```
<ord name="ABCD">
  <data>
```

```

        <select-or-insert name="ID" value="123456789"/>
    </data>
    <table name="Tabellenname">
        <data>
            <fields>
                <select-or-insert name="ID" value="123456789"/>
                <update name="name" value="ABSDIA"/>
            </fields>
        </data>
    </table>
</ord>

```

Tabellendaten können auf Ordner- (ord), Register- (reg) und Dokumentenebene (doc) eingefügt werden. Es werden dieselben Steueranweisungen, wie für die Indexdaten unterstützt.

Zusatz

Um zu verhindern, dass unbeabsichtigt Felder auf den Masken durch Importe geleert werden, weil z.B. ein Wert nicht übermittelt wird, berücksichtigt der Import keine leeren übergebenen Werte. Soll ein Feld explizit geleert werden, kann dies über das Schlüsselwort „empty“ (auf Kleinschreibung ist zu achten) erreicht werden.

Beispiel:

```

<update name="Referring Physician's Name" value="empty"/>

```

Import – OscCnImport

Einleitung

Die Komponente OscCnImport ist ein Konnektor (vgl. enaio® communicator-Dokumentation), der für den enaio® communicator ab Version 5.50 entwickelt wurde. Der Konnektor OscCnImport realisiert den gleichen Funktionsumfang, wie der Konnektor ConImpE. (siehe Seite 31)

Im Vergleich zu diesem besitzt er folgende Vorteile:

- Zugriff auf den Archiv-Server ohne Verwendung von Client-Komponenten
- Der Import startet sofort nach Starten des OSC-Servers ohne Wartezeit durch verzögerte Anmeldung am Archivserver

Mit OscCnImport.dll können Importe in die Datenbank des Archiv-Servers über den enaio® communicator realisiert werden. Es können Indexdaten und Dokumente importiert werden. Im Gegensatz zu den automatischen Aktionen, die zeit- bzw. ereignisgesteuert aufgerufen werden, sind Importe über den enaio® communicator nachrichtengesteuert. D.h., wird eine Importnachricht an den enaio® communicator gesendet, verarbeitet dieser diese Nachricht entsprechend seiner Konfiguration und sendet dann die Importdaten an OscCnImport. Die Struktur der Import-Daten muss dem Schema asimport.xsd entsprechen. Über OscCnImport können komplexe Strukturen in die Archivdatenbank importiert werden. Beispielsweise ist es möglich ein Importdokument zu erstellen, das Daten für mehrere Schränke, Register und Dokumente enthält. Das Import-Schema wird entsprechend der Releaseversion der Komponente gepflegt.

Importe in die Archivdatenbank, die über den enaio® communicator realisiert werden, können durch den Einsatz zusätzlicher Komponenten (in der Regel Transformer, die bestimmte Vor- und Nachbedingungen testen) sehr komplexe Logiken abbilden.

Voraussetzungen

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
OxSvrSpr.dll	OS	enaio® Serverzugriffsbibliothek
Oxidxdat.dll	OS	Admin-Komponente
Osccomdefs.dll	OS	enaio® communicator-Komponente
Axvbxml.dll	OS	enaio® communicator-Komponente
Mxml4.dll	Microsoft	
MS ADO	Microsoft	Ab Version 2.1

Für den Import werden Anmeldeinformationen eines Benutzers benötigt, in dessen Kontext der Import ausgeführt werden soll. Werden Konfigurationseinstellungen gewählt, die Zugriffe auf das Dateisystem benötigen (vgl. Konfiguration), ist darauf zu achten, dass der Import-Benutzer die ausreichenden Rechte besitzt.

Installation

Die Komponente OscCnImport wird standardmäßig im bin-Verzeichnis des Osc-Servers installiert. Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<connector name="conn_import" display="Import" definition="OscCnImport"
  configuration="Osc.Modules.xml">
  <comment>Import der Daten in das Archiv</comment>
  <input messagetype="mt_Import" supplier="tr_Pass"/>
</connector>
```

Außerdem setzt OscCnImport einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="conn_import">
  <!--mit oscpwdcrypter.exe verschlüsseltes Passwort -->
  <pwd>00000000000000</pwd>
  <!--Benutzer, in dessen Kontext der Import läuft -->
  <usr>HL7-Import</usr>
  <server>OSCServer#4000#100</server>
  <!--temporäres Verzeichnis (ab OS:4.2) -->
  <tmp-directory>D:\osc-dist\data\JiveX\out\preview-temp</tmp-directory>
  <!--sollen Import-Dokumente gespeichert werden (ab OS:4.2) -->
  <delete-import-documents>True</delete-import-documents>
  <!--Dokumente archivierbar setzen, Standard = False -->
  <archivable>>false</ archivable >
</module>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

```
server
pwd
usr
```

Für die Anmeldeinformationen ist ein Benutzer im System einzurichten, in dessen Kontext der Import läuft, beispielsweise „oscmport“.

pwd

Mit oscpwdcrypter.exe verschlüsseltes Passwort

usr

Benutzer, in dessen Kontext der Import läuft

server

Gibt eine Liste der zu verwendenden Server an. Die Syntax dafür lautet:

```
Rechnername#Port#Wahrscheinlichkeit für Benutzung des Servers in %
```

Bei nur einem verwendeten Server muss für die Wahrscheinlichkeit '100' angegeben werden.

tmp-directory

Die Einstellung tmp-directory wird benötigt, wenn zusätzlich zu den Indexdaten von Dokumenten Dokumente (z.B. XML, Word oder JPEG) importiert werden sollen. Diese werden in diesem Verzeichnis temporär gespeichert und nach dem Import wieder gelöscht.

delete-import-documents

Steuerung, ob das importierte Dateidokumente nach dem Import gelöscht werden soll oder nicht. Es wird darauf hingewiesen, dass dieses Attribut nur auf `true` gesetzt werden soll, wenn die zu importierenden Dateien nicht von anderen Systemen benötigt werden und die Berechtigung für das Löschen vorhanden ist.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Für die Erstellung des Importdokuments wird meistens eine XSLT-Transformation innerhalb des enaio® communicator realisiert.

Aufbau von Importdatensätzen

Der Aufbau der Importdatensätzen entspricht denen von OscConImpE. Siehe Seite 34.

Laborimport - oxvbcnli

Einleitung

Der Konnektor oxvbcnli.dll ist eine ActiveX-Dll. Dieser Konnektor importiert Labordaten via Serverjob in die Archivdatenbank. In der Regel werden die vom Labor gelieferte Daten HL7-Daten durch eine vorgeschaltete HL7 nach XML-Transformation und eine XSLT-Transformation in das Laborimportformat transformiert. Zusätzlich wird ein Transformer(osctrlookup.dll) benötigt, der in der Archivdatenbank eine Recherche durchführt, um sicherzustellen, dass Daten für den übermittelten Aufenthalt bereits in der Archivdatenbank erfasst worden sind. Ist dies nicht der Fall, wird ein Import von Labordaten abgelehnt. Der Konnektor übergibt die Daten an den Serverjob und dieser importiert die Daten auf Nachrichtenebene (die gesamte Nachricht) transaktionssicher in die Archivdatenbank. Optional kann per oxvbcnli ein Referenzdokument im Aufenthalt des Patienten angelegt werden. Der Hintergrund für dieses Feature ist, dass es nicht möglich ist anhand eines Aufenthalts zu erkennen, ob

es Laborwerte für die Anzeige im Laborviewer gibt. Wird ein Referenzdokument erstellt, kann sichergestellt werden, dass bei Vorhandensein des Dokuments auch Laborwerte für die Anzeige vorhanden sind. Die Konfigurationsangaben werden unter dem Punkt Konfiguration erläutert. Unter dem Punkt Voraussetzungen → Anlage des Referenzdokuments wird erläutert, welche Vorbereitungen in der Datenstruktur der Archivdatenbank erfüllt sein müssen.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Mxml4.dll	Microsoft	
Oxmljsc.Dll	enaio®	Server-Komponente
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Anlage des Referenzdokuments

Optional ist es möglich durch die Komponente ein Referenzdokument anzulegen. Dazu muss ein Index-Daten-Dokument (Siehe Bild) mit dem Namen **Labordaten-Referenz** angelegt werden. Auf diesem werden zwei Datenfelder, Anleger und Zeitpunkt erwartet.

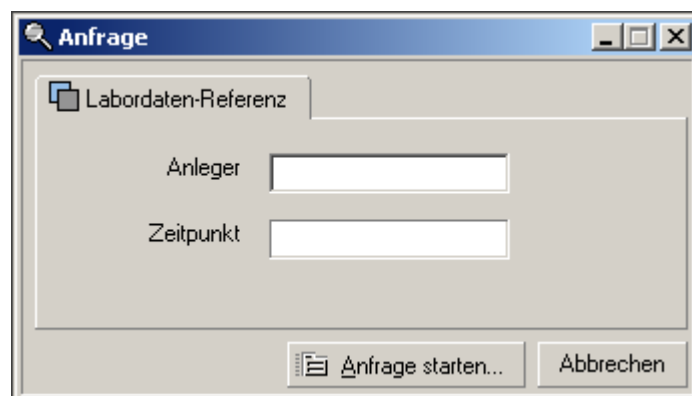


Abbildung 16

Installation

oxvbcnli.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<connector name="cnLabor"
           definition="OscComConnector"
           configuration="modules.xml">
  <input messageType="mtLabor" supplier="trLabor"/>
</connector>
```

Außerdem setzt oxvbcnli einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="cnLabor">
  <!-- Server-Anmeldung -->
  <usr>ROOT</usr>
  <!-- Passwort -->
  <pwd>HB01601611651521561421550000000000000000000000000</pwd>
  <!-- IP des Archiv-Servers -->
  <ip-address>127.0.0.1</ip-address>
  <!-- Port -->
  <tcp-port>4000</tcp-port>
  <!-- Temporäres Verzeichnis, wird für den Datenaustausch mit dem ArchivServer
benötigt, angebe optional-->
  <tmp-directory>D:\osc-dist\data\MIC\tmp</tmp-directory>
  <!-- Standard = false, wenn true, dann wird Referenzdokument erstellt.
Voraussetzungen müssen erfüllt sein, Siehe Voraussetzung-->
  <lab-ref-document>true</lab-ref-document>
  <register>Aufenthalt</register>
  <cabinet>Patient</cabinet>
  <ComProgId>oxvbcnli.Conn</ComProgId>
</module>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

ComProgId

usr

pwd

ip-address

tcp-port

Mit den Attributen `usr`, `pwd`, `ip-address`, `tcp-port` werden die Angaben für die Anmeldung an den Archiv-Server angegeben. Das Passwort muss kodiert angegeben werden. Dafür wird das Tool `OscPWDCrypter` verwendet.

Mit `tmp-directory` wird das temporäre Verzeichnis angegeben, das durch die Komponente für das Zwischenspeichern von Dateien verwendet wird.

Mit dem Attribut `lab-ref-document` wird gesteuert, ob ein Referenzdokument für den Laborimport angelegt werden soll. Standard ist `false`, wird `true` angegeben, können die Namen des Schrankes und des Registers, in dem das Dokument angelegt werden soll, angegeben werden. Als Standard wird „Patient“ für den Schrank und „Aufenthalt“ für das Register angenommen.

Ablauf des Laborimports

1. Daten empfangen (z.B. via TCP/IP oder Datei, in der Regel HL7)
2. Daten nach XML transformieren (in der Regel von HL7 nach XML)
3. Transformation der Daten in einen Anfragedatensatz, der durch `osctrlookup.dll` verarbeitet werden kann
4. Recherche via `osctrlookup.dll` in der Archivdatenbank
5. wenn Aufenthaltsdaten zu dem Labordatensatz gefunden wurden, Transformation in das Labordatenimport-Format
6. Import der Labordaten durch `oxvbcnli.dll` via Serverjob in die Archivdatenbank.

Aufbau der Importdatensätze

Der Import der Laborwerte wird über den Serverjob `med.ObservationInsert` realisiert. Dieser Job

erwartet ein bestimmtes Importformat. Dieses Format wird durch eine XSLT-Transformation erstellt. Es kann eine beliebige Anzahl von observations (Tag obr) mit einer beliebigen von results (Tag obx innerhalb von results) an die Komponente gesendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<med xmlns:os="http://www.optimal-systems.de/schemas"
      xmlns:med="http://www.optimal-systems.de/schemas/med"
      xmlns:loinc="http://www.optimal-systems.de/schemas/med/loinc">
  <patients>
    <patient id="113243">
      <visits>
        <visit id="113244">
          <requests transaction="12345">
            <obr placer-order-ident="000000657006"
                 filler-order-ident="RA01633322042003"
                 observation-dt="200304220858"
                 original="OBR||000000657006|RA01633322042003||||20030422095544|200304220858|||||200304220858|||||||20030422095544||K1|P">
              <results>
                <obx observation-ident="2951-2"
                     value-type-ident="ST"
                     value="134"
                     unit-ident="mmol/l"
                     ref-range="135-145"
                     result-changed-
                     dt="20030422095544"
                     original="OBX||ST|NA||134|mmol/l|135-145|"/>
              </results>
            </obr>
          </requests>
        </visit>
      </visits>
    </patient>
  </patients>
</med>
```

OxVbCnMrg

Einleitung

Der Konnektor OxVbCnMrg.dll ist eine ActiveX-Dll. Dieser Konnektor ist für das Mergen von Patienten (Patients) und Aufenthalten (Visits) zuständig. Wird durch das führende KIS eine ADT40 oder ADT42 Nachricht verschickt, so muss diese Nachricht an den Konnektor weitergeleitet werden. Der Konnektor kann weiterhin ab Version 5.50 die Nachrichten ADT24, ADT37, ADT45 und ab Version 6.0 die Nachrichten ADT46, ADT47 und ADT50 verarbeiten.

Voraussetzungen

Folgende Komponenten müssen auf dem System vorhanden sein.

False: Existiert das Quell- oder Zielobjekt nicht, dann wird das Mergen abgebrochen, aber kein Fehler generiert.

True: Existiert das Quell- oder Zielobjekt nicht, dann wird das Mergen abgebrochen und ein Fehler generiert.

```
-->
<no-source-or-target-is-error>False</no-source-or-target-is-error >
<!--
    Name der Komponente in der Registrierung
-->
<ComProgId>OxVbCnMrg.Conn</ComProgId>
</module>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

```
usr
pwd
ip-address
tcp-port
tmp-directory
ComProgId
```

Mit den Attributen `usr`, `pwd`, `ip-address`, `tcp-port` werden die Angaben für die Anmeldung an den Archiv-Server angegeben. Das Passwort muss kodiert angegeben werden. Dafür wird das Tool `OscPWDCrypter` verwendet.

Mit `tmp-directory` wird das temporäre Verzeichnis angegeben, das durch die Komponente für das Zwischenspeichern von Dateien verwendet wird.

Beispiele

ADT40

Um eine ADT40 Nachricht zu verarbeiten, muss folgendes Eingangs XML generiert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<merge>
  <message>ADT40</message>
  <cabinet>Patient</cabinet>
  <encounter id-name="Fallnummer">Aufenthalt</encounter>
  <target name="PatientenID" value="1" />
  <source name="PatientenID" value="2" />
</merge>
```

Im Tag `<message>` muss der Nachrichtentyp stehen der verarbeitet werden soll. In diesem Fall ADT40. Das Tag `<cabinet>` enthält den Schrank, in dem der Merge durchgeführt werden soll. Das Tag `<encounter>` setzt den Namen des Schlüsselfeldes für den Aufenthalt. Für den Fall, dass Aufenthalte mit gleichen Fallnummern existieren, werden lediglich die Dokumente verschoben und kein Fall mit der gleichen Fallnummer angelegt.

Das `name` Attribut im Tag `<source>` und `<target>` gibt an, nach welchem Feld auf der Maske der Patient ausgewählt werden soll. Im `value` Attribut steht der Wert des Feldes das in `name` beschrieben ist. Das XML bewirkt das Mergen von zwei Patienten. Der Patient mit der `PatientenID = 2` wird gemergt mit dem Patienten, der die `PatientenID = 1` hat. Der Wert des Feldes, das im `<source>` und `<target>` Tag angegeben ist, muss eindeutig sein, sonst kann der Merge nicht durchgeführt werden. Existiert die `PatientenID` des `targets` nicht, dann wird entsprechend der Einstellung des Parameters `change-id` verfahren.

Vorher	Nachher
Patient (PatientenID = 1) Aufenthalt (Fallnummer = 100) Aufenthalt (Fallnummer = 101)	Patient (PatientenID = 1) Aufenthalt (Fallnummer = 100) Aufenthalt (Fallnummer = 101)
Patient (PatientenID = 2) Aufenthalt (Fallnummer = 102)	Aufenthalt (Fallnummer = 102)

ADT42

Um eine ADT42 Nachricht zu verarbeiten, muss folgendes Eingangs XML generiert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<merge>
  <message>ADT42</message>
  <cabinet>Patient</cabinet>
  <register>Aufenthalt</register>
  <id name="PatientenID" value="1" />
  <target name="Fallnummer" value="100" />
  <source name="Fallnummer" value="101" />
</merge>
```

Im Tag `<message>` muss der Nachrichtentyp stehen, der verarbeitet werden soll. In diesem Fall ADT42. Das Tag `<cabinet>` enthält den Schrank, in dem der Merge durchgeführt werden soll. Das `<register>` Tag gibt an welches Register einem Visit entspricht, der gemergt werden soll.

Der Patient wird über das `<id>` Tag ermittelt, daß heißt dieses Feld muß eindeutig sein, sonst kann kein Merge durchgeführt werden. Im name Attribut steht der Name des Feldes anhand der Visit ausgewählt werden soll, im Attribut value der Wert des Feldes. Das XML bewirkt das mergen von zwei Aufenthalten im Patienten mit der PatientenID = 1. Der Aufenthalt mit der Fallnummer = 101 wird in den Aufenthalt mit der Fallnummer = 100 gemergt.

Vorher	Nachher
Patient (PatientenID = 1) Aufenthalt (Fallnummer = 100) Dokument1 Dokument2 Aufenthalt (Fallnummer = 101) Dokument3	Patient (PatientenID = 1) Aufenthalt (Fallnummer = 100) Dokument1 Dokument2 Dokument3

ADT45

Um eine ADT45 Nachricht zu verarbeiten, muss folgendes Eingangs XML generiert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<merge>
  <message>ADT45</message>
  <source>
    <cabinet name="Patient">
      <id name="PatientenID" value="2"/>
      <register name="Aufenthalt">
        <id name="Fallnummer" value="102"/>
      </register>
    </cabinet>
  </source>
  <target>
    <cabinet name="Patient">
      <id name="PatientenID" value="1"/>
      <register name="Aufenthalt">
```

```

        <id name="Fallnummer" value="100"/>
    </register>
</cabinet>
</target>
</merge>
    
```

Im Tag <message> muss der Nachrichtentyp stehen, der verarbeitet werden soll. In diesem Fall ADT45. Das Tag <source> enthält den Schrank und das Register, von dem aus ein Aufenthalt verschoben werden soll. Das <target> Tag enthält den Schrank und das Register, in den ein Aufenthalt verschoben werden soll.

Der Patient wird über das <id> Tag ermittelt, dass heißt dieses Feld muss eindeutig sein, sonst kann keine Verschiebung durchgeführt werden. Im name Attribut des <id> Tags steht der Name des Feldes, anhand dessen der Aufenthalt ausgewählt werden soll, im Attribut value der Wert des Feldes. Das XML bewirkt das Verschieben von einem Aufenthalt mit der Fallnummer 102 im Patienten mit der PatientenID = 2 zu einem Patienten mit der PatientenID=1. Das <id> Tag im target wird verwendet, damit das richtige Register gefunden werden kann.

Vorher	Nachher
Patient (PatientenID = 1)	Patient (PatientenID = 1)
Aufenthalt (Fallnummer = 100)	Aufenthalt (Fallnummer = 100)
Aufenthalt (Fallnummer = 101)	Aufenthalt (Fallnummer = 101)
Patient (PatientenID = 2)	Aufenthalt (Fallnummer = 102)
Aufenthalt (Fallnummer = 102)	Patient (PatientenID = 2)
Aufenthalt (Fallnummer = 103)	Aufenthalt (Fallnummer = 103)

ADT46/47

Um eine ADT46/47 Nachricht zu verarbeiten, muss folgendes Eingangs XML generiert werden:

```

<?xml version="1.0" encoding="UTF-8"?>
<merge>
    <message>ADT46</message>
    <cabinet>Patient</cabinet>
    <encounter id-name="Fallnummer">Aufenthalt</encounter>
    <target name="PatientenID" value="1" />
    <source name="PatientenID" value="2" />
</merge>
    
```

Im Tag <message> muss der Nachrichtentyp stehen, der verarbeitet werden soll. In diesem Fall ADT46 oder ADT47. Das Tag <cabinet> enthält den Schrank, in dem die Änderung durchgeführt werden soll.

Das name Attribut im Tag <source> und <target> gibt an, nach welchem Feld auf der Maske der Patient ausgewählt werden soll. Im value Attribut steht der Wert des Feldes, das in name beschrieben ist. Das XML bewirkt eine Änderung der Patienten-ID. Der Patient mit der bisherigen PatientenID = 1 erhält die neue Patienten-ID 2. Der Wert des Feldes, das im <source> und <target> Tag angegeben ist, muss eindeutig sein, sonst kann der Merge nicht durchgeführt werden.

Vorher	Nachher
Patient (PatientenID = 1)	Patient (PatientenID = 2)
Aufenthalt (Fallnummer = 100)	Aufenthalt (Fallnummer = 100)
Aufenthalt (Fallnummer = 101)	Aufenthalt (Fallnummer = 101)
Aufenthalt (Fallnummer = 102)	Aufenthalt (Fallnummer = 102)

ADT50

Um eine ADT50 Nachricht zu verarbeiten, muss folgendes Eingangs XML generiert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<merge>
  <message>ADT50</message>
  <cabinet>Patient</cabinet>
  <register>Aufenthalt</register>
  <id name="PatientenID" value="1" />
  <target name="Fallnummer" value="100" />
  <source name="Fallnummer" value="111" />
</merge>
```

Im Tag <message> muss der Nachrichtentyp stehen, der verarbeitet werden soll. In diesem Fall ADT50. Das Tag <cabinet> enthält den Schrank, in dem der Change durchgeführt werden soll. Das <register> Tag gibt an welches Register einem Visit entspricht, der geändert werden soll.

Der Patient wird über das <id> Tag ermittelt, daß heißt dieses Feld muß eindeutig sein, sonst kann kein Change durchgeführt werden. Im name Attribut steht der Name des Feldes, anhand dessen der Patient ausgewählt werden soll, im Attribut value der Wert des Feldes. Das XML bewirkt, dass die Fallnummer im <target> Tag durch die Fallnummer im <source> Tag ersetzt wird.

Vorher	Nachher
Patient (PatientenID = 1)	Patient (PatientenID = 1)
Aufenthalt (Fallnummer = 100)	Aufenthalt (Fallnummer = 111)
Aufenthalt (Fallnummer = 101)	Aufenthalt (Fallnummer = 101)
Aufenthalt (Fallnummer = 102)	Aufenthalt (Fallnummer = 102)

Bemerkungen

Nachrichten vom Typ ADT41, ADT43 werden von dieser Komponente nicht unterstützt, da im System keine Accounts abgebildet werden. Sollen diese Nachrichten verarbeitet werden, kann dies mit der Importkomponente OscConImpE.exe abgebildet werden, da diese Nachrichten im System als Updates auf Ordner oder Registerebene, eines Feldes (z.B. PatientenID, Fallnummer), abgebildet werden können.

OscReadFile

Einleitung

Die Komponente OscReadFile ist ein Konnektor (vgl. enaio® communicator-Dokumentation), der für den enaio® communicator entwickelt und als ActiveX-DLL realisiert wurde.

Mit OscReadFile kann der Inhalt einer Datei zeilenweise an nachfolgende Komponenten für die weitere Verarbeitung weitergegeben werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

Die Komponente OscreadFile.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein. Diese Komponente wird im bin-Verzeichnins des enaio® communicator abgelegt.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<connector name="cnreadfile"
           definition="OscComConnector"
           configuration="modules.xml">
  <input messagetype="mt_read" supplier="trimport"/>
</connector>
```

Außerdem OscreadFile.Dll setzt einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<osc>
  <module name="cnreadfile">
    <!--Filter auf zu lesende Dateien -->
    <filenames>*.csv</filenames>
    <!--Verzeichnis, in dem die Komponente Dateien lesen soll -->
    <directory>c:\readfile</directory>
    <!-- ProgId der Komponente -->
    <ComProgId>OscReadFile.Conn</ComProgId>
  </module>
</osc>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

filenames

directory

ComProgId

Mit dem Tag `filenames` wird bestimmt, welche Dateien gelesen werden sollen. Die Maske wird in der Regel mit *.Dateiendung angegeben.

Die Einstellung `directory` wird angegeben, um zu bestimmen, in welchem Verzeichnis die Komponente Dateien liest.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Anwendungsbeispiel

Dieses Beispiel basiert auf der Konfiguration in dem Abschnitt Konfiguration.

Dateninput: Die Datei „mustermann.csv“ (vgl. Bild 12) hat folgenden Inhalt

Mustermann;Manfred;50;männlich

Musterfrau;Gerda;50;weiblich

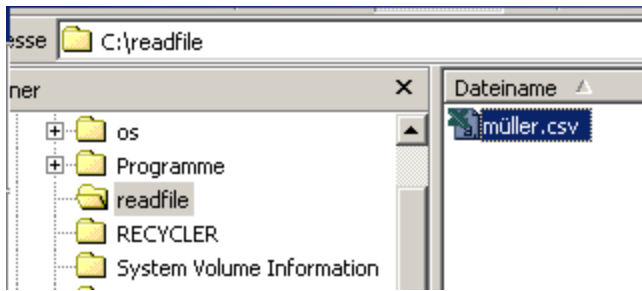


Abbildung 17

Diese Datei würde von OscReadFile zeilenweise an die Komponenten, die den Nachrichtentyp „mt_read“ abonniert haben (vgl. Dafür enaio® communicator-Dokumentation), weitergereicht werden. Nach erfolgreicher Verarbeitung, d.h. wenn alle Zeilen eingelesen und weitergereicht worden sind, verschiebt die Komponente die eingelesene Datei ins Tagesverzeichnis (vgl. Bild 13) unterhalb des Verzeichnis readfile/correct. Treten Fehler bei der Verarbeitung auf, wird die Datei in ein Tagesverzeichnis unterhalb des Verzeichnis readfile/error verschoben.

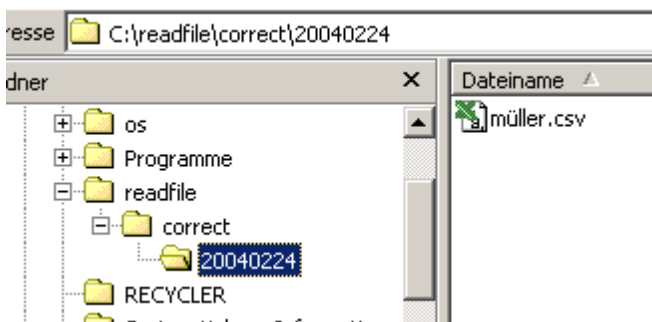


Abbildung 18

OscCnNull

Einleitung

Die Komponente OscCnNull ist ein Konnektor (vgl. enaio® communicator-Dokumentation), der für den enaio® communicator entwickelt und als ActiveX-DLL realisiert wurde.

Mit OscCnNull kann für Testzwecke verwendet werden. Dieser Konnektor implementiert keine Geschäftslogik und funktioniert lediglich als Datensenke (Daten werden jedoch nicht gespeichert). In der Anlaufphase eines Projektes kann diese Komponente beispielsweise als Ersatz für OscConImpE verwendet werden. Wenn die Kommunikationspartner das Datenformat abgestimmt haben, kann OscConImpE eingesetzt werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

Die Komponente OscCnNull.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<connector name="cnNull"
  definition="OscComConnector"
  configuration="modules.xml">
  <input messageType="mt_import" supplier="trimport"/>
</connector>
```

Außerdem OscConImpE setzt einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<osc>
  <module name="cnNull">
    <!--ist im Standard auf false gestellt, hiermit kann gesteuert werden, ob die
Komponente die Verarbeitung als wahr oder falsch bewertet -->
    <return-false>true</ return-false >
    <!-- ProgId der Komponente -->
    <ComProgId>OscCnNull.Conn</ComProgId>
  </module>
</osc>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

return-false

ComProgId

Die Einstellung return-false wird angegeben, um zu bestimmen, ob die Verarbeitung als wahr oder falsch gekennzeichnet wird.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

OscCnMSMQ

Die Komponente OscCnMSMQ ist ein Konnektor (vgl. enaio® communicator-Dokumentation), der für den enaio® communicator entwickelt und als COM-DLL realisiert wurde.

Mit OscCnMSMQ können Nachrichten in die Microsoft Message Queue gestellt werden und von dort wieder abgeholt werden. Das Einsatzszenario ergibt sich für Lösungen, in denen der enaio® communicator dafür verantwortlich ist Nachrichten persistent in einem sicheren Datenspeicher zu sichern oder eignet sich für den Datenaustausch mit Applikationen, die Nachrichten an den enaio® communicator senden oder empfangen sollen.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Folgende Komponenten müssen auf dem System vorhanden sein. Die Komponente ist mit dem MS .NET-Framework entwickelt worden. Daher muss die .NET-Runtime auf dem Installationsrechner vorhanden sein.

Abhängigkeiten		
Interop.OscComDefs.Dll	enaio®	enaio® communicator-

		Komponente
Osccomdefs.dll	OS	enaio® communicator-Komponente

Installation

Die Komponente OscCnMSMQ.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Da es sich um eine COM-Dll (.NET) muss diese mit dem Tool regasm.exe (Bestandteil des .NET SDK) registriert werden.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

Für das Anlegen der Queue gibt es zwei Möglichkeiten.

- Der Administrator erstellt per Microsoft-Management-Konsole die Queue und diese wird im enaio® communicator konfiguriert.
- In der Konfiguration wird die Queue angegeben und OscCnMSMQ versucht diese Queue zu erzeugen. Das gilt jedoch nur für den Sende-Modus.

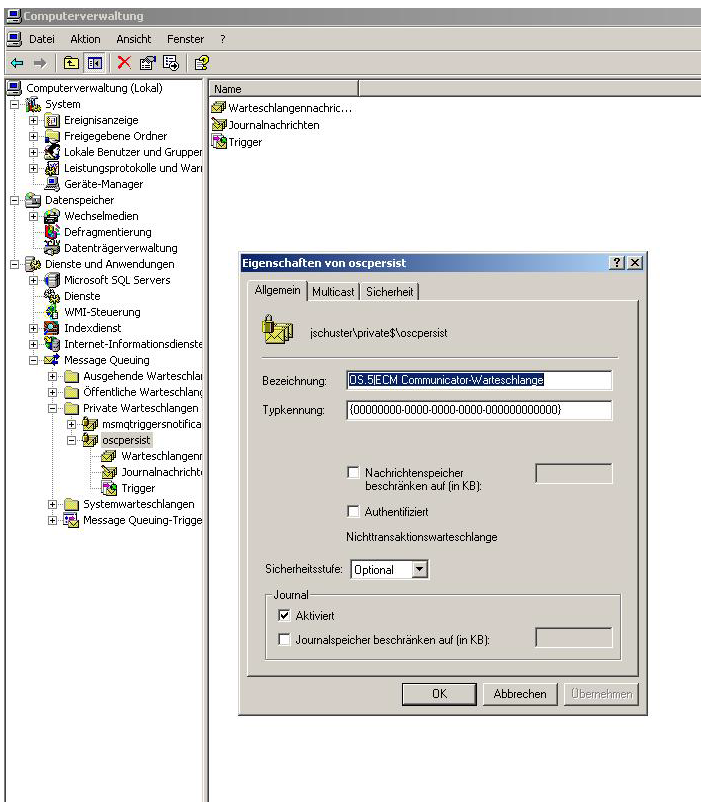


Abbildung 19

1. Hier als Konnektor, der Nachrichten zum Senden an die MSMQ erhält.

```
<connector name="cnMSMQ"
           definition="OscComConnector"
           configuration="modules.xml">
  <input messageType="mt_HI7" supplier="trHI7"/>
</connector>
```

2. Hier als Konnektor, der Nachrichten von der MSMQ holt.


```

<connector name="cnMSMQ"
            definition="OscComConnector"
            configuration="modules.xml">
  <output message="mt_MSMQ"/>
</connector>

```

Außerdem OscConImpE setzt einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

Receive-Modus

```

<osc>
  <module name="cnMSMQ">
    <!-- als Sender wird hier „send“ eingetragen -->
    <!-- send oder receive -->
    <mode>receive</mode>
    <!-- Fehler Queue, nur wenn der Konnekteur im Receive-Modus -->
    <!-- konfiguriert wurde -->
    <!-- Wenn dieses Attribut konfiguriert, wird im -->
    <!-- Fehlerfall die Nachricht auf dieses Queue transferiert -->
    <error-queue>.\private$\oscError</error-queue>
    <!-- Name der Warteschlange wird in der Konsole angezeigt -->
    <!-- Bezeichnung der Queue -->
    <label>Bezeichnung der Queue</label>
    <!-- Name der Warteschlange wird in der Konsole angezeigt -->
    <queue>rechner\private$\oscpersist</queue>
    <!-- ProgId der Komponente -->
    <!-- Timeout in Sekunden -->
    <timeout>0,5</timeout>
    <!-- ProgId der Komponente -->
    <ComProgId>OscCnMSMQ.Conn</ComProgId>
  </module>
</osc>

```

Wird keine error-queue konfiguriert, wird im Falle einer nicht erfolgreichen Verarbeitung der Nachricht diese trotzdem von der Queue genommen und ist damit verloren.

Send-Modus

```

<osc>
  <module name="cnMSMQ">
    <!-- als Sender wird hier „send“ eingetragen -->
    <!-- send oder receive -->
    <mode>send</mode>
    <!-- Name der Warteschlange wird in der Konsole angezeigt -->
    <!-- Bezeichnung der Queue -->
    <label>Bezeichnung der Queue</label>
    <!-- Name der Warteschlange wird in der Konsole angezeigt -->
    <queue>rechner\private$\oscpersist</queue>
    <!-- ProgId der Komponente -->
    <!-- ProgId der Komponente -->
    <ComProgId>OscCnMSMQ.Conn</ComProgId>
  </module>
</osc>

```

```
</module>
</osc>
```

Sonstiges

Wird für die Message-Queue vom Administrator als ein Journal angelegt, werden vorhandene Nachrichten aus der Message-Queue beim Herunterfahren des Rechners durch Windows in das Journal verschoben. Damit kann ein persistenter Datenspeicher für die Zwischenspeicherung von Nachrichten eingerichtet werden. Um diese Nachrichten nachher von der Message-Queue abzuholen, muss die Queue für den Konnektor entsprechend konfiguriert werden. Hier z.B.

```
<queue>rechner\private$\oscpersist\Journal$</queue>
```

Weitergehende Informationen zu MSMQ können der Fachliteratur bzw. im Internet gefunden werden. Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

OscCnOsEMail

Die Komponente OscCnOsEMail ist ein Konnektor, der für den enaio® communicator entwickelt und als COM-DLL realisiert wurde.

Mit OscCnOsEMail können via EMail an konfigurierte Empfänger versendet werden. Beispielsweise lässt sich diese Komponente dann einsetzen, wenn der Administrator über bestimmte Vorfälle im Nachrichtenfluss benachrichtigt werden soll.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Oxmljsc.dll	enaio®	enaio® Komponente
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

Die Komponente OscCnOsEMail.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration und in der Modulkonfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen. Die Funktionalität dieses Konnektors kann nur genutzt werden, wenn im System ein Mailserver eingerichtet wurde.


```
<file-extension>Achtung</email-subject>
```

```
</osc>
```

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

HL7XML-Transformer – OscHl7XmlCTrans/OscHl7XmlTrans

Einleitung

Die Komponenten OscHl7XmlCTrans und OscHl7XmlTrans sind Transformer die HL7-Daten (vgl. HL7-Spezifikation) in HL7-XML transformiert. Dieses XML-Format orientiert sich weitestgehend an den XML Encoding Syntax aus der HL7-Spezifikation Version 2. Durch die Transformation der HL7-Daten nach XML wird die weitere Verarbeitung vereinfacht und Komponenten können standardisiert (z.B. per Xpath, XSLT usw.) auf die HL7-Daten zugreifen.

Installation

Die Komponente OscHl7XmlCTrans.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. OscConSocket ist Bestandteil jeder enaio® communicator-Installation.

Konfiguration

Für die Konfiguration der Komponente OscHl7XmlCTrans werden HL7-Kenntnisse vorausgesetzt. Werden Einstellungen benötigt, die von Standardkonfigurationen abweichen, müssen die entsprechenden Konfigurationsdateien (module und HL7-Grammatik) angepasst werden.

Unterschied OscHl7XmlCTrans und OscHl7XmlTrans

Mit dem Transformer OscHl7XmlTrans steht ein HL7-Transformer zur Verfügung, der standardkonform mit nicht bekannten Feldern innerhalb von Segmenten umgehen kann, wenn diese in der HL7-Grammatik nicht angegeben worden sind. Die unbekannt Felder werden übergangen und ab einem neuen Segment werden die Daten wieder transformiert.

Der Transformer OscHl7XmlTrans verwendet den Standard-Namensraum für HL7-Dokumente Version zwei. Es besteht die Möglichkeit einen restriktiven Modus zu konfigurieren. In diesem Fall wird als Segmenttrenner nur ein Carriage Return (ASCII-Zeichen 13) akzeptiert. Als Encoding Set wird ASCII oder 8859/1 akzeptiert. Wird ein anderes Encoding Set angegeben, wird die Nachricht als fehlerhaft gekennzeichnet.

```
<ADT_A03 xmlns="urn:hl7-org:v2xml">
```

Damit wird es notwendig mit diesem Namensraum in Stylesheets zu arbeiten.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc"
```

```
xmlns:date="http://exslt.org/dates-and-times"
```

```
xmlns:hl7="urn:hl7-org:v2xml">
```

Anschließend kann der Prefix in XPath-Anweisungen verwendet werden, um auf Daten aus dem XML zu zugreifen.

```
<update name="Patienten-Nr." value="{hl7:PID/hl7:PID.3[1]/hl7:CX.1}"/>
```

Allerdings ist es auch möglich, den Parser zu veranlassen, keinen Namensraum zu verwenden (Siehe Konfiguration).

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen

werden sollen.

```
<rule name="rhl7xml"
      transformer="C"
      definition="OscHl7XmlCTrans "
      initfile="modules.xml">
  <input  message="mthl7"/>
  <output message="mthl7xml"/>
</rule>
```

Oder

```
<rule name="rhl7xml"
      transformer="C"
      definition="OscHl7XmlTrans"
      initfile="modules.xml">
  <input  message="mthl7"/>
  <output message="mthl7xml"/>
</rule>
<transformation name="trhl7xml"
                rule="rhl7xml">
  <input  supplier="cnhl7" message="mthl7"/>
  <output message="mthl7xml"/>
</transformation>
```

Außerdem setzt OscHl7XmlCTrans einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus.

```
<module name="rhl7xml">
  <Hl7ToXmlConverter>
    <!--restriktiver Modus ab OS.520 ->
    <!--Namesraum ab OS.520 Standard = true ->
    <HL7>
      <strict-mode>true</ strict-mode >
      <use-hl7-v2-namespace>>false</use-hl7-v2-namespace>
    </HL7>
    <dictionary>HL7_2_3.DAT</dictionary>
    <logfile>HL7XMLConversion.log</logfile>
    <output>.\HL7.xml</output>
    <xsl-Transformation>testxsl.xsl</xsl-Transformation>
    <!-- muss an verwendete Version angepasst werden -->
    <structures>
      <!-- version 2.3.1 -->
      <event msg="ADT_A01" struct="ADT_A01"/>
      <event msg="ADT_A02" struct="ADT_A02"/>
      <event msg="ADT_A03" struct="ADT_A03"/>
      <event msg="ADT_A04" struct="ADT_A01"/>
      <event msg="ADT_A05" struct="ADT_A01"/>
      <event msg="ADT_A06" struct="ADT_A01"/>
      <event msg="ADT_A07" struct="ADT_A01"/>
      <event msg="ADT_A08" struct="ADT_A01"/>
      <event msg="ADT_A09" struct="ADT_A09"/>
      <event msg="ADT_A10" struct="ADT_A09"/>
      <event msg="ADT_A11" struct="ADT_A01"/>
      <event msg="ADT_A12" struct="ADT_A02"/>
      <event msg="ADT_A13" struct="ADT_A03"/>
      <event msg="ADT_A14" struct="ADT_A01"/>
    </structures>
  </Hl7ToXmlConverter>
</module>
```

```
<event msg="ADT_A15" struct="ADT_A09"/>
<event msg="ADT_A16" struct="ADT_A16"/>
<event msg="ADT_A17" struct="ADT_A17"/>
<event msg="ADT_A18" struct="ADT_A18"/>
<event msg="ADT_A20" struct="ADT_A20"/>
<event msg="ADT_A21" struct="ADT_A02"/>
<event msg="ADT_A22" struct="ADT_A02"/>
<event msg="ADT_A23" struct="ADT_A02"/>
<event msg="ADT_A24" struct="ADT_A24"/>
<event msg="ADT_A25" struct="ADT_A02"/>
<event msg="ADT_A26" struct="ADT_A02"/>
<event msg="ADT_A27" struct="ADT_A02"/>
<event msg="ADT_A28" struct="ADT_A28"/>
<event msg="ADT_A29" struct="ADT_A02"/>
<event msg="ADT_A30" struct="ADT_A30"/>
<event msg="ADT_A31" struct="ADT_A01"/>
<event msg="ADT_A32" struct="ADT_A02"/>
<event msg="ADT_A33" struct="ADT_A02"/>
<event msg="ADT_A34" struct="ADT_A30"/>
<event msg="ADT_A35" struct="ADT_A30"/>
<event msg="ADT_A36" struct="ADT_A30"/>
<event msg="ADT_A37" struct="ADT_A37"/>
<event msg="ADT_A38" struct="ADT_A38"/>
<event msg="ADT_A39" struct="ADT_A39"/>
<event msg="ADT_A40" struct="ADT_A39"/>
<event msg="ADT_A41" struct="ADT_A39"/>
<event msg="ADT_A42" struct="ADT_A39"/>
<event msg="ADT_A43" struct="ADT_A43"/>
<event msg="ADT_A44" struct="ADT_A43"/>
<event msg="ADT_A45" struct="ADT_A45"/>
<event msg="ADT_A46" struct="ADT_A30"/>
<event msg="ADT_A47" struct="ADT_A30"/>
<event msg="ADT_A48" struct="ADT_A30"/>
<event msg="ADT_A49" struct="ADT_A30"/>
<event msg="ADT_A50" struct="ADT_A50"/>
<event msg="ADT_A51" struct="ADT_A50"/>
</structures>
<structures_2.3>
  <!-- version 2.3.1 -->
  <event msg="ADT_A01" struct="ADT_A01"/>
  <event msg="ADT_A02" struct="ADT_A02"/>
  <event msg="ADT_A03" struct="ADT_A03"/>
  <event msg="ADT_A04" struct="ADT_A01"/>
  <event msg="ADT_A05" struct="ADT_A01"/>
  <event msg="ADT_A06" struct="ADT_A06"/>
  <event msg="ADT_A07" struct="ADT_A01"/>
  <event msg="ADT_A08" struct="ADT_A01"/>
  <event msg="ADT_A09" struct="ADT_A09"/>
  <event msg="ADT_A10" struct="ADT_A09"/>
```

```
<event msg="ADT_A11" struct="ADT_A09"/>
<event msg="ADT_A12" struct="ADT_A12"/>
<event msg="ADT_A13" struct="ADT_A01"/>
<event msg="ADT_A14" struct="ADT_A01"/>
<event msg="ADT_A15" struct="ADT_A09"/>
<event msg="ADT_A16" struct="ADT_A16"/>
<event msg="ADT_A17" struct="ADT_A17"/>
<event msg="ADT_A18" struct="ADT_A18"/>
<event msg="ADT_A20" struct="ADT_A20"/>
<event msg="ADT_A21" struct="ADT_A02"/>
<event msg="ADT_A22" struct="ADT_A02"/>
<event msg="ADT_A23" struct="ADT_A02"/>
<event msg="ADT_A24" struct="ADT_A24"/>
<event msg="ADT_A25" struct="ADT_A02"/>
<event msg="ADT_A26" struct="ADT_A02"/>
<event msg="ADT_A27" struct="ADT_A02"/>
<event msg="ADT_A28" struct="ADT_A28"/>
<event msg="ADT_A29" struct="ADT_A02"/>
<event msg="ADT_A30" struct="ADT_A30"/>
<event msg="ADT_A31" struct="ADT_A01"/>
<event msg="ADT_A32" struct="ADT_A02"/>
<event msg="ADT_A33" struct="ADT_A02"/>
<event msg="ADT_A34" struct="ADT_A30"/>
<event msg="ADT_A35" struct="ADT_A30"/>
<event msg="ADT_A36" struct="ADT_A30"/>
<event msg="ADT_A37" struct="ADT_A37"/>
<event msg="ADT_A38" struct="ADT_A38"/>
<event msg="ADT_A39" struct="ADT_A39"/>
<event msg="ADT_A40" struct="ADT_A39"/>
<event msg="ADT_A41" struct="ADT_A39"/>
<event msg="ADT_A42" struct="ADT_A39"/>
<event msg="ADT_A43" struct="ADT_A43"/>
<event msg="ADT_A44" struct="ADT_A43"/>
<event msg="ADT_A45" struct="ADT_A45"/>
<event msg="ADT_A46" struct="ADT_A30"/>
<event msg="ADT_A47" struct="ADT_A30"/>
<event msg="ADT_A48" struct="ADT_A30"/>
<event msg="ADT_A49" struct="ADT_A30"/>
<event msg="ADT_A50" struct="ADT_A50"/>
<event msg="ADT_A51" struct="ADT_A50"/>
</structures_2.3>
<structures_2.4>
  <!-- version 2.4 -->
  <event msg="ADT_A01" struct="ADT_A01"/>
  <event msg="ADT_A02" struct="ADT_A02"/>
  <event msg="ADT_A03" struct="ADT_A03"/>
  <event msg="ADT_A04" struct="ADT_A01"/>
  <event msg="ADT_A05" struct="ADT_A05"/>
  <event msg="ADT_A06" struct="ADT_A06"/>
```

```
<event msg="ADT_A07" struct="ADT_A01"/>
<event msg="ADT_A08" struct="ADT_A01"/>
<event msg="ADT_A09" struct="ADT_A09"/>
<event msg="ADT_A10" struct="ADT_A09"/>
<event msg="ADT_A11" struct="ADT_A09"/>
<event msg="ADT_A12" struct="ADT_A09"/>
<event msg="ADT_A13" struct="ADT_A01"/>
<event msg="ADT_A14" struct="ADT_A05"/>
<event msg="ADT_A15" struct="ADT_A15"/>
<event msg="ADT_A16" struct="ADT_A16"/>
<event msg="ADT_A17" struct="ADT_A17"/>
<event msg="ADT_A18" struct="ADT_A18"/>
<event msg="ADT_A20" struct="ADT_A20"/>
<event msg="ADT_A21" struct="ADT_A21"/>
<event msg="ADT_A22" struct="ADT_A21"/>
<event msg="ADT_A23" struct="ADT_A21"/>
<event msg="ADT_A24" struct="ADT_A24"/>
<event msg="ADT_A25" struct="ADT_A21"/>
<event msg="ADT_A26" struct="ADT_A21"/>
<event msg="ADT_A27" struct="ADT_A21"/>
<event msg="ADT_A28" struct="ADT_A05"/>
<event msg="ADT_A29" struct="ADT_A21"/>
<event msg="ADT_A30" struct="ADT_A30"/>
<event msg="ADT_A31" struct="ADT_A05"/>
<event msg="ADT_A32" struct="ADT_A21"/>
<event msg="ADT_A33" struct="ADT_A21"/>
<event msg="ADT_A34" struct="ADT_A30"/>
<event msg="ADT_A35" struct="ADT_A30"/>
<event msg="ADT_A36" struct="ADT_A30"/>
<event msg="ADT_A37" struct="ADT_A37"/>
<event msg="ADT_A38" struct="ADT_A38"/>
<event msg="ADT_A39" struct="ADT_A39"/>
<event msg="ADT_A40" struct="ADT_A39"/>
<event msg="ADT_A41" struct="ADT_A39"/>
<event msg="ADT_A42" struct="ADT_A39"/>
<event msg="ADT_A43" struct="ADT_A43"/>
<event msg="ADT_A44" struct="ADT_A43"/>
<event msg="ADT_A45" struct="ADT_A45"/>
<event msg="ADT_A46" struct="ADT_A30"/>
<event msg="ADT_A47" struct="ADT_A30"/>
<event msg="ADT_A48" struct="ADT_A30"/>
<event msg="ADT_A49" struct="ADT_A30"/>
<event msg="ADT_A50" struct="ADT_A50"/>
<event msg="ADT_A51" struct="ADT_A50"/>
</structures_2.4>
</HI7ToXmlConverter>
</module>
```

Hier Mit diesen Angaben wird die Abbildung von HL7-Nachrichten zu einem bestimmten Ereignis

(z.B. ADT^A02 – Verlegung) auf eine definierte HL7-XML-Struktur (ADT_A02-Schema) festgelegt. Diese Abbildung unterscheidet sich geringfügig je nach HL7-Version, dementsprechend ist die Datei ggf. auf die HL7-Version anzupassen.

dictionary

Hier muss das die Datei mit der zu verwendeten Grammatik angegeben werden (vgl. HL7-Grammatik). Diese Grammatik unterscheidet sich zwischen den verschiedenen HL7-Versionen.

Logfile

Hier wird der Name des Logfiles angegeben. Dieses Logfile dient nur dem Zweck der Fehlersuche. Als Wurzelverzeichnis wird das logs-Verzeichnis in der enaio® communicator-Installation verwendet. Daher wird hier lediglich der zu verwendende Dateiname angegeben. In diesem Logfile wird jeweils der letzte Parser-Durchlauf ausführlich protokolliert, vor Verarbeitung einer neuen Nachricht wird das Logfile jeweils wieder gelöscht.

structures

Für jede zu unterstützende HL7-Version wird ein eigener structures-Block angegeben. Derzeitig werden structures für die HL7-Versionen 2.3 und 2.4 angegeben. Unter event wird in msg der Typ und das Ereignis der HL7-Nachricht in der Notation Typ_Event angegeben. In struct wird angegeben welche abstrakte HL7-Struktur verwendet werden soll. Für weitergehende Informationen wird an dieser Stelle auf die HL7-spezifikation verwiesen (vgl. z.B. HL7 Table 0354 im Kapitel 2 Version 2.4).

HL7-Grammatik

Die zu verwendene Grammatik der Komponente wird standardmäßig in einer dat-Datei hinterlegt. Verwiesen wird auf diese Datei innerhalb der module-Konfiguration (vgl. dictionary).

In dieser Datei werden die Segmente, Felder und Komponenten, die von der Komponente verarbeitet werden sollen, beschrieben. Diese Grammatik ist ähnlich einem XML-Schema zu verstehen, aber wesentlich kompakter aufgebaut.

Für die verschiedenen HL7-Versionen existieren angepasste Versionen dieser Dateien. Die Generierung der Dateien erfolgt mit geeigneten Skripten (Perl) anhand eines DB-Exports aus der HL7-Grammatik-Datenbank (Frank Oemig, siehe <http://www.hl7.de>).

Im Einzelfall kann eine nachträgliche manuelle Anpassung der Grammatikdatei erfolgen. Dies kann z.B. dann erforderlich sein, wenn zusätzlich vereinbarte Segmente (Z-Segmente) verarbeitet werden sollen, oder wenn die Struktur der zu verarbeitenden Nachrichten vom HL7-Standard abweicht. *Solche geänderten Grammatik-Dateien sind unbedingt mit einem eigenen, projektspezifischen Dateinamen zu versehen!!* Die Anpassungen sollten detailliert dokumentiert werden.

Beispiel anhand von ADT_A01

```
ADT_A01
(MSH, EVN, PID, PD1?, NK1*, PV1, PV2?, DB1*, OBX*, AL1*, DG1*, DRG?, ADT_A01.PROCEDURE*
, GT1*, ADT_A01.INSURANCE*, ACC?, UB1?, UB2?, NK1*, DG1*, ZBE?)
```

Mit diesem Eintrag in der Grammatik wird bestimmt aus welchen Segmenten eine ADT_A01-Nachricht bestehen kann. Es können die Platzhalter * (beliebige Anzahl) und ? (optional 0 oder 1) verwendet werden. Wird kein Platzhalter verwendet ist es ein Pflichtsegment.

Das MSH-Segment besteht aus folgenden Feldern.

```
MSH
(Mvgl.1, Mvgl.2, Mvgl.3?, Mvgl.4?, Mvgl.5?, Mvgl.6?, Mvgl.7?, Mvgl.8?, Mvgl.9, Mvgl.
10, Mvgl.11, Mvgl.12, Mvgl.13?, Mvgl.14?, Mvgl.15?, Mvgl.16?, Mvgl.17?, Mvgl.18*, Mv
gl.19?, Mvgl.20?)
```

Die einzelnen Felder werden ebenfalls in der Grammatik beschrieben.

```
Mvgl.1 %ST Item='1' Table='0' LongName='Field Separator' Type='ST'
Mvgl.2 %ST Item='2' Table='0' LongName='Encoding Characters' Type='ST'
Mvgl.3 %HD Item='3' Table='361' LongName='Sending Application' Type='HD'
Mvgl.4 %HD Item='4' Table='362' LongName='Sending Facility' Type='HD'
```

```
Mvgl.5 %HD Item='5' Table='361' LongName='Receiving Application' Type='HD'
Mvgl.6 %HD Item='6' Table='362' LongName='Receiving Facility' Type='HD'
Mvgl.7 %TS Item='7' Table='0' LongName='Date/Time Of Message' Type='TS'
Mvgl.8 %ST Item='8' Table='0' LongName='Security' Type='ST'
Mvgl.9 %CM_MSG_TYPE Item='9' Table='76' LongName='Message Type'
Type='CM_MSG_TYPE'
Mvgl.10 %ST Item='10' Table='0' LongName='Message Control ID' Type='ST'
Mvgl.11 %PT Item='11' Table='0' LongName='Processing ID' Type='PT'
Mvgl.12 %VID Item='12' Table='104' LongName='Version ID' Type='VID'
Mvgl.13 %NM Item='13' Table='0' LongName='Sequence Number' Type='NM'
Mvgl.14 %ST Item='14' Table='0' LongName='Continuation Pointer' Type='ST'
Mvgl.15 %ID Item='15' Table='155' LongName='Accept Acknowledgment Type'
Type='ID'
Mvgl.16 %ID Item='16' Table='155' LongName='Application Acknowledgment
Type' Type='ID'
Mvgl.17 %ID Item='17' Table='0' LongName='Country Code' Type='ID'
Mvgl.18 %ID Item='692' Table='211' LongName='Character Set' Type='ID'
Mvgl.19 %CE Item='693' Table='0' LongName='Principal Language Of Message'
Type='CE'
Mvgl.20 %ID Item='1317' Table='356' LongName='Alternate Character Set
Handling Scheme' Type='ID'
```

Nach der Angabe des Feldes wird mit %XX der Datentyp angegeben. Für die zu verwendenden Datentypen wird hier auf die HL7-Spezifikation verwiesen (vgl. Data Types in Version 2.4 Kapitel 2.17).

Beispiel

In diesem Beispiel wird eine HL7-Nachricht und der entsprechende Output der Komponente gezeigt.
HL7-Nachricht.

```
MSH|^~\&|PROSIGHT|PROSIGHT|||200308261014^YYYYMMDDHHMM||ADT^A08|20030826101
4533|P|2.4|||NE|NE|D
EVN|A08|20030826101449^YYYYMMDDHHMMSS||5
PID|1|188123^^^&PROSIGHT&L|188123^^^&PROSIGHT&L||Mustermann&^Manfred^^^^||1
9610923^YYYYMMDD|M||Musterstr.
14^^Halle^^06114^D^H~^^^^^Deutschland^N|||||ke|||||||D|
NK1|1|Mustermann^^^^^|ef|Musterstr. 14^^Halle^^06114^D^H|||||fr|||
PV1|1|I|3A&Station
3A&20030826101300&^^^TRA&Traumatologie&20030826101300&^|||||1507^Bastian&
Holger^^^^^Dr. med.^H~1507^Musterfrau&^Gerda^^^^^Dr.
med.^V|||||1507||0|||287084^^^&PROSIGHT&L^&PROSIGHT&L^20030826101300&YYYYM
MDDHHMMSS^|||||20030826101300^YYYYMMDDHHMMSS|||||||D|||2003082610
1300^YYYYMMDDHHMMSS|||||186380^^&PROSIGHT&L^&PROSIGHT&L^20030826101300&YY
YYMMDDHHMMSS^
PV2|||||||
OBX|||8345-1^Koerpergewicht^LN||0|g^^ISO+|||||200308260000^YYYYMMDDHHMM
IN1|1||101097008^^^^NIIP~1097008^^^^NII~100001^^^^AND|AOK Halle|Robert-
Franz-Ring 16^^Halle^^06108^D^B|||||
|||||||N|||||
IN2|||||GK
ZVK|1|AOK SACHSEN-
ANHALT|1097008|85101|221962166|1000|9|200512310000^YYYYMMDDHHMM||Giest|Step
han||196109230000^YYYYMMDDHHMM|Rembrandtstr.
14||06114|Halle|J|N|200308210000^YYYYMMDDHHMM|
```

Folgendes HL7-XML wurde daraus generiert.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ADT_A01>
  <MSH>
```

```

<Mvgl.1>|</Mvgl.1>
<!-- Item='1' Table='0' LongName='Field Separator' Type='ST' -->
<Mvgl.2>^~\&amp;</Mvgl.2>
<!-- Item='2' Table='0' LongName='Encoding Characters' Type='ST' -->
<Mvgl.3>
  <HD.1>PROSIGHT</HD.1>
  <!-- LongName='namespace ID' Type='IS' Table='300' -->
</Mvgl.3>
<Mvgl.4>
  <HD.1>PROSIGHT</HD.1>
  <!-- LongName='namespace ID' Type='IS' Table='300' -->
</Mvgl.4>
<Mvgl.7>200308261014</Mvgl.7>
<!-- Item='7' Table='0' LongName='Date/Time Of Message' Type='TS' -->
<Mvgl.8>YYYYMMDDHHMM</Mvgl.8>
<!-- Item='8' Table='0' LongName='Security' Type='ST' -->
<Mvgl.10>ADT</Mvgl.10>
<!-- Item='10' Table='0' LongName='Message Control ID' Type='ST' -->
<Mvgl.11>
  <PT.1>A08</PT.1>
  <!-- LongName='processing ID' Type='ID' Table='103' -->
</Mvgl.11>
<Mvgl.12>
  <VID.1>200308261014533</VID.1>
  <!-- LongName='version ID' Type='ID' Table='104' -->
</Mvgl.12>
<Mvgl.13>P</Mvgl.13>
<!-- Item='13' Table='0' LongName='Sequence Number' Type='NM' -->
<Mvgl.14>2.4</Mvgl.14>
<!-- Item='14' Table='0' LongName='Continuation Pointer' Type='ST' -->
<Mvgl.17>NE</Mvgl.17>
<!-- Item='17' Table='0' LongName='Country Code' Type='ID' -->
<Mvgl.18>NE</Mvgl.18>
<!-- Item='692' Table='211' LongName='Character Set' Type='ID' -->
<Mvgl.19>
  <CE.1>D</CE.1>
  <!-- LongName='identifier' Type='ST' Table='0' -->
</Mvgl.19>
</MSH>
<EVN>
  <EVN.1>A08</EVN.1>
  <!-- Item='99' Table='3' LongName='Event Type Code' Type='ID' -->
  <EVN.2>20030826101449</EVN.2>
  <!-- Item='100' Table='0' LongName='Recorded Date/Time' Type='TS' -->
  <EVN.3>YYYYMMDDHHMMSS</EVN.3>
  <!-- Item='101' Table='0' LongName='Date/Time Planned Event' Type='TS' -->
  <EVN.5>
    <XCN.1>5</XCN.1>
    <!-- LongName='ID number (ST)' Type='ST' Table='0' -->
  </EVN.5>
</EVN>
<PID>
  <PID.1>1</PID.1>
  <!-- Item='104' Table='0' LongName='Set ID - PID' Type='SI' -->
  <PID.2>
    <CX.1>188123</CX.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX.4>
      <HD.2>PROSIGHT</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
      <HD.3>L</HD.3>
      <!-- LongName='universal ID type' Type='ID' Table='301' -->
    </CX.4>
  </PID.2>
</PID>

```

```

    </CX.4>
  </PID.2>
  <PID.3>
    <CX.1>188123</CX.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX.4>
      <HD.2>PROSIGHT</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
      <HD.3>L</HD.3>
      <!-- LongName='universal ID type' Type='ID' Table='301' -->
    </CX.4>
  </PID.3>
  <PID.5>
    <XPN.1>
      <CM.1>Mustermann</CM.1>
      <!-- LongName='&lt;segment ID' Type='ST' Table='0' -->
    </XPN.1>
    <XPN.2>Manfred</XPN.2>
    <!-- LongName='given name' Type='ST' Table='0' -->
  </PID.5>
  <PID.7>19610923</PID.7>
  <!-- Item='110' Table='0' LongName='Date/Time Of Birth' Type='TS' -->
  <PID.8>YYYYMMDD</PID.8>
  <!-- Item='111' Table='1' LongName='Sex' Type='IS' -->
  <PID.9>
    <XPN.1>
      <CM.1>M</CM.1>
      <!-- LongName='&lt;segment ID' Type='ST' Table='0' -->
    </XPN.1>
  </PID.9>
  <PID.12>Musterstr. 14</PID.12>
  <!-- Item='115' Table='289' LongName='County Code' Type='IS' -->
  <PID.13>
    <XTN.2>Halle</XTN.2>
    <!-- LongName='telecommunication use code' Type='ID' Table='201' -->
    <XTN.4>06114</XTN.4>
    <!-- LongName='Email address' Type='ST' Table='0' -->
    <XTN.5>D</XTN.5>
    <!-- LongName='Country Code' Type='NM' Table='0' -->
    <XTN.6>H</XTN.6>
    <!-- LongName='Area/city code' Type='NM' Table='0' -->
  </PID.13>
  <PID.13>
    <XTN.6>Deutschland</XTN.6>
    <!-- LongName='Area/city code' Type='NM' Table='0' -->
    <XTN.7>N</XTN.7>
    <!-- LongName='Phone number' Type='NM' Table='0' -->
  </PID.13>
  <PID.19>ke</PID.19>
  <!-- Item='122' Table='0' LongName='SSN Number - Patient' Type='ST' -->
  <PID.28>
    <CE.1>D</CE.1>
    <!-- LongName='identifier' Type='ST' Table='0' -->
  </PID.28>
</PID>
<NK1>
  <NK1.1>1</NK1.1>
  <!-- Item='190' Table='0' LongName='Set ID - NK1' Type='SI' -->
  <NK1.2>
    <XPN.1>
      <CM.1>Mustermann</CM.1>
      <!-- LongName='&lt;segment ID' Type='ST' Table='0' -->

```

```

    </XPN.1>
  </NK1.2>
  <NK1.3>
    <CE.1>ef</CE.1>
    <!-- LongName='identifier' Type='ST' Table='0' -->
  </NK1.3>
  <NK1.4>
    <XAD.1>Musterstr. 14</XAD.1>
    <!-- LongName='street address' Type='ST' Table='0' -->
    <XAD.3>Halle</XAD.3>
    <!-- LongName='city' Type='ST' Table='0' -->
    <XAD.6>06114</XAD.6>
    <!-- LongName='country' Type='ID' Table='0' -->
    <XAD.7>D</XAD.7>
    <!-- LongName='address type' Type='ID' Table='190' -->
    <XAD.8>H</XAD.8>
    <!-- LongName='other geographic designation' Type='ST' Table='0' -->
  </NK1.4>
  <NK1.10>fr</NK1.10>
  <!-- Item='199' Table='0' LongName='Next of Kin / Associated Parties Job Title' Type='ST' -->
</NK1>
<PV1>
  <PV1.1>1</PV1.1>
  <!-- Item='131' Table='0' LongName='Set ID - PV1' Type='SI' -->
  <PV1.2>I</PV1.2>
  <!-- Item='132' Table='4' LongName='Patient Class' Type='IS' -->
  <PV1.3>
    <PL.1>3A</PL.1>
    <!-- LongName='point of care' Type='IS' Table='302' -->
    <PL.2>Station 3A</PL.2>
    <!-- LongName='room' Type='IS' Table='303' -->
    <PL.3>20030826101300</PL.3>
    <!-- LongName='bed' Type='IS' Table='304' -->
    <PL.7>TRA</PL.7>
    <!-- LongName='building' Type='IS' Table='307' -->
    <PL.8>Traumatologie</PL.8>
    <!-- LongName='floor' Type='IS' Table='308' -->
    <PL.9>20030826101300</PL.9>
    <!-- LongName='Location description' Type='ST' Table='0' -->
  </PV1.3>
  <PV1.11>
    <PL.1>1507</PL.1>
    <!-- LongName='point of care' Type='IS' Table='302' -->
    <PL.2>Musterfrau</PL.2>
    <!-- LongName='room' Type='IS' Table='303' -->
    <PL.4>
      <HD.1>Gerda</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
    </PL.4>
    <PL.9>Dr. med.</PL.9>
    <!-- LongName='Location description' Type='ST' Table='0' -->
  </PV1.11>
  <PV1.12>H</PV1.12>
  <!-- Item='142' Table='87' LongName='Preadmit Test Indicator' Type='IS' -->
  <PV1.13>1507</PV1.13>
  <!-- Item='143' Table='92' LongName='Re-admission Indicator' Type='IS' -->
  <PV1.14>Bastian</PV1.14>
  <!-- Item='144' Table='23' LongName='Admit Source' Type='IS' -->
  <PV1.16>Holger</PV1.16>
  <!-- Item='146' Table='99' LongName='VIP Indicator' Type='IS' -->
  <PV1.17>
    <XCN.5>Dr. med.</XCN.5>

```

```
<!-- LongName='suffix (e.g., JR or III)' Type='ST' Table='0' -->
<XCN.6>V</XCN.6>
<!-- LongName='prefix (e.g., DR)' Type='ST' Table='0' -->
</PV1.17>
<PV1.22>1507</PV1.22>
<!-- Item='152' Table='45' LongName='Courtesy Code' Type='IS' -->
<PV1.24>0</PV1.24>
<!-- Item='154' Table='44' LongName='Contract Code' Type='IS' -->
<PV1.27>287084</PV1.27>
<!-- Item='157' Table='0' LongName='Contract Period' Type='NM' -->
<PV1.31>PROSIGHT</PV1.31>
<!-- Item='161' Table='21' LongName='Bad Debt Agency Code' Type='IS' -->
<PV1.32>L</PV1.32>
<!-- Item='162' Table='0' LongName='Bad Debt Transfer Amount' Type='NM' -->
<PV1.34>PROSIGHT</PV1.34>
<!-- Item='164' Table='111' LongName='Delete Account Indicator' Type='IS' -->
<PV1.35>L</PV1.35>
<!-- Item='165' Table='0' LongName='Delete Account Date' Type='DT' -->
<PV1.37>
  <CM.1>20030826101300</CM.1>
  <!-- LongName='&lt;segment ID' Type='ST' Table='0' -->
  <CM.2>YYYYMMDDHHMMSS</CM.2>
  <!-- LongName='sequence' Type='NM' Table='0' -->
</PV1.37>
<PV1.43>
  <PL.1>20030826101300</PL.1>
  <!-- LongName='point of care' Type='IS' Table='302' -->
  <PL.2>YYYYMMDDHHMMSS</PL.2>
  <!-- LongName='room' Type='IS' Table='303' -->
</PV1.43>
</PV1>
<PV2>
  <PV2.1>
    <PL.7>D</PL.7>
    <!-- LongName='building' Type='IS' Table='307' -->
  </PV2.1>
  <PV2.2>
    <CE.1>20030826101300</CE.1>
    <!-- LongName='identifier' Type='ST' Table='0' -->
    <CE.2>YYYYMMDDHHMMSS</CE.2>
    <!-- LongName='text' Type='ST' Table='0' -->
  </PV2.2>
  <PV2.3>
    <CE.2>186380</CE.2>
    <!-- LongName='text' Type='ST' Table='0' -->
    <CE.5>PROSIGHT</CE.5>
    <!-- LongName='alternate text' Type='ST' Table='0' -->
    <CE.6>L</CE.6>
    <!-- LongName='name of alternate coding system' Type='ST' Table='0' -->
  </PV2.3>
  <PV2.4>
    <CE.2>PROSIGHT</CE.2>
    <!-- LongName='text' Type='ST' Table='0' -->
    <CE.3>L</CE.3>
    <!-- LongName='name of coding system' Type='ST' Table='0' -->
    <CE.5>20030826101300</CE.5>
    <!-- LongName='alternate text' Type='ST' Table='0' -->
    <CE.6>YYYYMMDDHHMMSS</CE.6>
    <!-- LongName='name of alternate coding system' Type='ST' Table='0' -->
  </PV2.4>
  <PV2.6>PV2</PV2.6>
  <!-- Item='186' Table='0' LongName='Patient Valuables Location' Type='ST' -->
```

```

<PV2.14>OBX</PV2.14>
<!-- Item='715' Table='0' LongName='Previous Service Date' Type='DT' -->
<PV2.17>8345-1</PV2.17>
<!-- Item='718' Table='0' LongName='Purge Status Date' Type='DT' -->
<PV2.18>Koerpergewicht</PV2.18>
<!-- Item='719' Table='214' LongName='Special Program Code' Type='IS' -->
<PV2.19>LN</PV2.19>
<!-- Item='720' Table='136' LongName='Retention Indicator' Type='ID' -->
<PV2.21>0</PV2.21>
<!-- Item='722' Table='215' LongName='Visit Publicity Code' Type='IS' -->
<PV2.22>g</PV2.22>
<!-- Item='723' Table='136' LongName='Visit Protection Indicator' Type='ID' -->
<PV2.23>
  <XON.2>ISO+</XON.2>
  <!-- LongName='organization name type code' Type='IS' Table='204' -->
</PV2.23>
<PV2.26>200308260000</PV2.26>
<!-- Item='727' Table='0' LongName='Previous Treatment Date' Type='DT' -->
<PV2.27>YYYYMMDDHHMM</PV2.27>
<!-- Item='728' Table='112' LongName='Expected Discharge Disposition' Type='IS' -->
<PV2.28>IN1</PV2.28>
<!-- Item='729' Table='0' LongName='Signature on File Date' Type='DT' -->
<PV2.29>1</PV2.29>
<!-- Item='730' Table='0' LongName='First Similar Illness Date' Type='DT' -->
<PV2.30>
  <CE.2>101097008</CE.2>
  <!-- LongName='text' Type='ST' Table='0' -->
  <CE.6>NIIP</CE.6>
  <!-- LongName='name of alternate coding system' Type='ST' Table='0' -->
</PV2.30>
<PV2.31>1097008</PV2.31>
<!-- Item='732' Table='219' LongName='Recurring Service Code' Type='IS' -->
<PV2.35>NII</PV2.35>
<!-- Item='736' Table='136' LongName='Military Non-Availability Code' Type='ID' -->
<PV2.36>100001</PV2.36>
<!-- Item='737' Table='136' LongName='Newborn Baby Indicator' Type='ID' -->
</PV2>
<DB1>
  <DB1.3>
    <CX.1>AND</CX.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX.2>AOK Halle</CX.2>
    <!-- LongName='check digit' Type='NM' Table='0' -->
    <CX.3>Robert-Franz-Ring 16</CX.3>
    <!-- LongName='code identifying the check digit scheme employed' Type='ID' Table='61' -->
    <CX.4>
      <HD.2>Halle</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
    </CX.4>
    <CX.5>06108</CX.5>
    <!-- LongName='identifier type code' Type='IS' Table='203' -->
    <CX.6>
      <HD.1>D</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
      <HD.2>B</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
    </CX.6>
  </DB1.3>
</DB1>
<OBX>
  <OBX.16>
    <XCN.4>N</XCN.4>

```

```

        <!-- LongName='middle initial or name' Type='ST' Table='0' -->
    </OBX.16>
</OBX>
<ADT_A01.INSURANCE>
    <IN2>
        <IN2.5>GK</IN2.5>
        <!-- Item='476' Table='137' LongName='Mail Claim Party' Type='IS' -->
    </IN2>
</ADT_A01.INSURANCE>
</ADT_A01>

```

XSLT - OscXsltTransService

Einleitung

Die Komponente OscXsltTransservice (oscxsllttransservice.dll) ist ein Transformer der XML-Daten mittels XSL-Stylesheet transformiert.

Installation

Die Komponente OscXsltTransservice.Dll wird im bin-Verzeichnis der enaio® communicator-Installation installiert. OscConFile ist Bestandteil jeder enaio® communicator-Installation.

Konfiguration

OscPassThrough wird nur in der aktiven Konfiguration konfiguriert. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

```

<rule name="rl_xslt" transformer="XSLT" definition="import-mapping.xml">
    <input messageType="mt_hl7_xml"/>
    <output messageType="mt_imp_xml"/>
</rule>
<transformation name="tr_xslt" rule="rl_xslt">
    <input supplier="cnin" messageType="mt_hl7_xml"/>
    <output messageType="mt_imp_xml"/>
</transformation>

```

Um die Komponente OscXsltTransservice einzubinden, muss im Attribut transformer „XSLT“ angegeben werden. In der definition wird das zu verwendene Stylesheet für die Transformation angegeben. Für weitergehende Informationen zur Konfiguration wird an dieser Stelle auf die Dokumentation zur Konfiguration des enaio® communicators verwiesen.

Zusätzlich kann der Transformer OscXsltTransformer mit einer Zeichenkette initialisiert werden.

Dafür steht das Attribut `initstring` zur Verfügung.

Bis Version 5.20 können die Einträge `split` oder `remove-linefeed` verwendet werden.

```

<rule name="rl_xslt" transformer="XSLT" definition="import-mapping.xml" initstring="remove-linefeed">
<rule name="rl_xslt" transformer="XSLT" definition="import-mapping.xml" initstring="split">

```

Mit `remove-linefeed` wird erzwungen, dass als Zeilenumbruch nur ein Carriage Return verwendet wird. Dieses ist beispielsweise zwingend notwendig, wenn Sie HL7-Daten mittels XSLT generieren möchten.

Mit `split` wird erzwungen, dass alle Kindsknoten des Wurzelknoten, des transformierten XML-Dokumentes einzeln versendet werden.

Die Einstellungen `remove-linefeed` und `split` schließen sich gegenseitig aus.

Zusätzlich steht der Eintrag `filewatcher` zur Verfügung.

```

<rule name="rl_xslt" transformer="XSLT" definition="import-mapping.xml" initstring="remove-linefeed,filewatcher">

```



```
<rule name="rl_xslt" transformer="XSLT" definition="import-mapping.xml" initstring="split,filewatcher">
```

Mit `filewatcher` wird erzwungen, dass das durch `OscXsltTransformer` geladene Stylesheet überwacht wird. Wenn dieses editiert wurde, wird das Stylesheet neu geladen (mit einer zeitlichen Verzögerung bis zu 60 Sekunden). Falls das Laden fehlschlägt, lädt die Komponente das vorher als Backup gesicherte Stylesheet. Dem Protokoll kann entnommen werden, ob dieser Prozess erfolgreich ausgeführt wurde. Dadurch kann das Beenden mit anschließendem Neustart der `enaio® communicator` wegen kleinen Änderungen in Stylesheets vermieden werden. Allerdings sollten die Stylesheets nur von fachkundigem Personal geändert werden.

Beispiele

ASCII-Import

Im ersten Beispiel soll eine HL7-Nachricht, die im HL7-XML-Format zur Verfügung steht, in das Importformat für den ASCII-Import transformiert werden.

Die Beispielnachricht sieht folgend aus:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ACK_P01>
  <MSH>
    <Mvgl.1>|</Mvgl.1>
    <!-- Item='1' Table='0' LongName='Field Separator' Type='ST' -->
    <Mvgl.2>^~\&amp;</Mvgl.2>
    <!-- Item='2' Table='0' LongName='Encoding Characters' Type='ST' -->
    <Mvgl.5>
      <HD.1>OSEPA_1MED</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
    </Mvgl.5>
    <Mvgl.6>
      <HD.1>EPA</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
    </Mvgl.6>
    <Mvgl.7>20040305101500</Mvgl.7>
    <!-- Item='7' Table='0' LongName='Date/Time Of Message' Type='TS' -->
    <Mvgl.9>
      <CM_MSG_TYPE.1>ACK</CM_MSG_TYPE.1>
      <!-- LongName='message type' Type='ID' Table='76' -->
    </Mvgl.9>
    <Mvgl.10>834</Mvgl.10>
    <!-- Item='10' Table='0' LongName='Message Control ID' Type='ST' -->
    <Mvgl.11>
      <PT.1>T</PT.1>
      <!-- LongName='processing ID' Type='ID' Table='103' -->
    </Mvgl.11>
    <Mvgl.12>
      <VID.1>2.4</VID.1>
      <!-- LongName='version ID' Type='ID' Table='104' -->
    </Mvgl.12>
    <Mvgl.15>NE</Mvgl.15>
    <!-- Item='15' Table='155' LongName='Accept Acknowledgment Type' Type='ID' -->
    <Mvgl.16>NE</Mvgl.16>
    <!-- Item='16' Table='155' LongName='Application Acknowledgment Type' Type='ID' -->
  </MSH>
  <MSA>
    <MSA.1>AR</MSA.1>
    <!-- Item='18' Table='8' LongName='Acknowledgment Code' Type='ID' -->
    <MSA.2>1568073299.200402051</MSA.2>
    <!-- Item='10' Table='0' LongName='Message Control ID' Type='ST' -->
    <MSA.3>ORACLE error 20035--ORA-20035: Mind. ein Diagnose-Segment konnte nicht
    verarbeitet werden:--DG1!1!ICDGM2004!B96.2 * Pneumonie durch Enterobacter!Enterobacter als
    Ursache von Krkh. sonst klassifiz.!20040216084014!FD!!!!!!!1.3!Petra Duhm-
```

Harbeck!!!!1568073306FD1 OSEPA_1MED 1--Diagnose-Schlüssel: B96.2 Keinen gültigen ICD-Code (ggf. inkl. passendem Kennz. !+*) im DB-Katalog gefunden!--ORA-06512: at "CKIS_MGR.PG_COMDIS_DIAGTHERA", line 731--ORA-06512: at "CKIS_MGR.COMDIS_BAR_IMP", line 122--ORA-06512: at line 3--</MSA.3>

```

<!-- Item='20' Table='0' LongName='Text Message' Type='ST' -->
</MSA>
<ERR>
  <ERR.1>
    <CM.4>
      <CE.1>ORACLE error 20035--ORA-20035: Mind. ein Diagnose-Segment konnte nicht
      verarbeitet werden:--DG1!1!ICDGM2004!B96.2 * Pneumonie durch Enterobacter!Enterobacter als
      Ursache von Krkh. sonst klassifiz.!20040216084014!FD!!!!!!!!!!1.3!Petra Duhm-
      Harbeck!!!!1568073306FD1 OSEPA_1MED 1--Diagnose-Schlüssel: B96.2 Keinen gültigen ICD-Code
      (ggf. inkl. passendem Kennz. !+*) im DB-Katalog gefunden!--ORA-06512: at
      &quot;CKIS_MGR.PG_COMDIS_DIAGTHERA&quot;, line 731--ORA-06512: at
      &quot;CKIS_MGR.COMDIS_BAR_IMP&quot;, line 122--ORA-06512: at line 3--</CE.1>
    <!-- LongName='identifizier' Type='ST' Table='0' -->
  </CM.4>
</ERR.1>
</ERR>
</ACK_P01>

```

Die Daten, die mit dieser Nachricht transportiert werden, sollen auf ein Dokument in dem Register Aufenthalt unter dem Ordner Patient importiert werden. Die Patienten- und Aufenthaltsnummer ist unbekannt. Auf Dokumentenebene gibt es einen eindeutigen Schlüssel, der in der Nachricht in dem Segment MSA.2 (gelb markiert) hinterlegt ist. Außerdem soll der Status der Nachricht anhand der Information im Segment MSA.1 nur eine Beschreibung aus MSA.3 extrahiert werden.

Das Stylesheet für diese Transformation kann beispielsweise so aussehen.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  xmlns:osc="urn:optimal-systems-de:osc"
  xmlns:date="http://exslt.org/dates-and-times">
  <xsl:output method="text" indent="no" omit-xml-declaration="no" encoding="ISO-8859-1"/>

```

```

<!--
#####
#####
#      Javascript
#####
#####
-->

```

```

<msxsl:script language="JavaScript" implements-prefix="osc"><![CDATA[

```

```

function getVld(val)
{
  var ret = "";
  if (val.match(/[0-9]/.[0-9]/))
  {
    var values = val.split(".");
    return (values[1]);
  }
  return ret;
}

```

```

function getVStatus(val)
{
    var ret = 0;
    switch (val)
    {
// Nachricht wurde verarbeitet ACK
        case "AA":
            ret = 2;
            break;
        case "CA":
            ret = 2;
            break;
// Fehler NACK
        case "AE":
            ret = 3;
            break;
        case "CE":
            ret = 3;
            break;
// Fehler NACK
        case "AR":
            ret = 3;
            break;
        case "CR":
            ret = 3;
    }
    return ret;
}

function getOslId(val)
{
    var values = val.split(".");
    return (values[0]);
}

```

```
]]></msxsl:script>
```

```

<xsl:template match="/">
    <xsl:variable name="Msg_Type" select="*/MSH/Mvgl.9/CM_MSG_TYPE.1"/>
    <xsl:choose>
        <xsl:when test="$Msg_Type='ACK'"> <!-- Empfangende Applikation hat
Nachricht akzeptiert -->
            <xsl:apply-templates select="/*" mode="ACK"/>
        </xsl:when>
        <xsl:when test="$Msg_Type='NACK'"> <!-- Empfangende Applikation hat
Nachricht nicht akzeptiert -->
            <xsl:apply-templates select="/*" mode="NACK"/>
        </xsl:when>
    <xsl:otherwise>

```

```

        <xsl:text>Diese Nachricht wird nicht verarbeitet! (</xsl:text>
        <xsl:value-of select="$Msg_Type"/>
        <xsl:text>)</xsl:text>
    </xsl:otherwise>
</xsl:choose>
</xsl:template>
<xsl:template match="*" mode="ACK">
    <xsl:text>Pat-ID C(40);A-ID C(50);V-ID C(10);V-Status C(50);V-OsId C(16);V-Descr
C(128)</xsl:text>
    <xsl:text>&#xD;&#xA;</xsl:text>
    <xsl:value-of select="0"/><xsl:text>;</xsl:text>
    <xsl:value-of select="0"/><xsl:text>;</xsl:text>
    <xsl:value-of select="osc:getVId(string(/*/MSA/MSA.2))"/><xsl:text>;</xsl:text>
    <xsl:value-of select="osc:getVStatus(string(/*/MSA/MSA.1))"/><xsl:text>;</xsl:text>
    <xsl:value-of select="osc:getOsId(string(/*/MSA/MSA.2))"/><xsl:text>;</xsl:text>
    <xsl:value-of select="*/MSA/MSA.3"/>
    <xsl:text>&#xD;&#xA;</xsl:text>
</xsl:template>
</xsl:stylesheet>

```

OscConImpE-Import

Im zweiten Beispiel soll ein einfaches Mapping von HL7-XML für den Import über die enaio® communicator-Komponente OscConImpE.Exe erstellt werden.

Die Beispielnachricht (A01 nach HL7-Standard) sieht folgend aus:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<ADT_A01>
  <MSH>
    <Mvgl.1>|</Mvgl.1>
    <!-- Item='1' Table='0' LongName='Field Separator' Type='ST' -->
    <Mvgl.2>^~\&amp;</Mvgl.2>
    <!-- Item='2' Table='0' LongName='Encoding Characters' Type='ST' -->
    <Mvgl.3>
      <HD.1>PROSIGHT</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
    </Mvgl.3>
    <Mvgl.4>
      <HD.1>PROSIGHT</HD.1>
      <!-- LongName='namespace ID' Type='IS' Table='300' -->
    </Mvgl.4>
    <Mvgl.7>
      <TS.1>200308261155</TS.1>
      <!-- LongName='date/time' Type='ST' Table='0' -->
      <TS.2>YYYYMMDDHHMM</TS.2>
      <!-- LongName='degree of precision' Type='ST' Table='0' -->
    </Mvgl.7>
    <Mvgl.9>
      <CM_MSG.1>ADT</CM_MSG.1>
      <!-- LongName='message type' Type='ID' Table='76' -->
      <CM_MSG.2>A01</CM_MSG.2>
      <!-- LongName='trigger event' Type='ID' Table='3' -->
    </Mvgl.9>
    <Mvgl.10>200308261155383</Mvgl.10>
    <!-- Item='10' Table='0' LongName='Message Control ID' Type='ST' -->
    <Mvgl.11>
      <PT.1>P</PT.1>
      <!-- LongName='processing ID' Type='ID' Table='103' -->

```

```

</Mvgl.11>
<Mvgl.12>
  <VID.1>2.4</VID.1>
  <!-- LongName='version ID' Type='ID' Table='104' -->
</Mvgl.12>
<Mvgl.15>NE</Mvgl.15>
<!-- Item='15' Table='155' LongName='Accept Acknowledgment Type' Type='ID' -->
<Mvgl.16>NE</Mvgl.16>
<!-- Item='16' Table='155' LongName='Application Acknowledgment Type' Type='ID' -->
<Mvgl.17>D</Mvgl.17>
<!-- Item='17' Table='399' LongName='Country Code' Type='ID' -->
</MSH>
<EVN>
  <EVN.1>A01</EVN.1>
  <!-- Item='99' Table='3' LongName='Event Type Code' Type='ID' -->
  <EVN.2>
    <TS.1>20030826115535</TS.1>
    <!-- LongName='date/time' Type='ST' Table='0' -->
    <TS.2>YYYYMMDDHHMMSS</TS.2>
    <!-- LongName='degree of precision' Type='ST' Table='0' -->
  </EVN.2>
  <EVN.4>1</EVN.4>
  <!-- Item='102' Table='62' LongName='Event Reason Code' Type='IS' -->
</EVN>
<PID>
  <PID.1>1</PID.1>
  <!-- Item='104' Table='0' LongName='Set ID - PID' Type='SI' -->
  <PID.2>
    <CX.1>140603</CX.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX.4>
      <HD.2>PROSIGHT</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
      <HD.3>L</HD.3>
      <!-- LongName='universal ID type' Type='ID' Table='301' -->
    </CX.4>
  </PID.2>
  <PID.3>
    <CX.1>140603</CX.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX.4>
      <HD.2>PROSIGHT</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
      <HD.3>L</HD.3>
      <!-- LongName='universal ID type' Type='ID' Table='301' -->
    </CX.4>
  </PID.3>
  <PID.5>
    <XPN.1>
      <FN.1>Mustermann</FN.1>
      <!-- LongName='surname' Type='ST' Table='0' -->
    </XPN.1>
    <XPN.2>Manfred</XPN.2>
    <!-- LongName='given name' Type='ST' Table='0' -->
  </PID.5>
  <PID.7>
    <TS.1>19870219</TS.1>
    <!-- LongName='date/time' Type='ST' Table='0' -->
    <TS.2>YYYYMMDD</TS.2>
    <!-- LongName='degree of precision' Type='ST' Table='0' -->
  </PID.7>
  <PID.8>M</PID.8>

```

```

<!-- Item='111' Table='1' LongName='Administrative Sex' Type='IS' -->
<PID.11>
  <XAD.1>
    <SAD.1>Musterstr. 33</SAD.1>
    <!-- LongName='street or mailing address' Type='ST' Table='0' -->
  </XAD.1>
  <XAD.3>Musterstadt</XAD.3>
  <!-- LongName='city' Type='ST' Table='0' -->
  <XAD.5>06682</XAD.5>
  <!-- LongName='zip or postal code' Type='ST' Table='0' -->
  <XAD.7>H</XAD.7>
  <!-- LongName='address type' Type='ID' Table='190' -->
</PID.11>
<PID.11>
  <XAD.7>N</XAD.7>
  <!-- LongName='address type' Type='ID' Table='190' -->
</PID.11>
<PID.14>
  <XTN.1>1</XTN.1>
  <!-- LongName='[(999)] 999-9999 [X99999][C any text]' Type='TN' Table='0' -->
</PID.14>
<PID.26>
  <CE.1>D</CE.1>
  <!-- LongName='identifier (ID)' Type='ID' Table='0' -->
</PID.26>
</PID>
<PV1>
  <PV1.1>1</PV1.1>
  <!-- Item='131' Table='0' LongName='Set ID - PV1' Type='SI' -->
  <PV1.2>O</PV1.2>
  <!-- Item='132' Table='4' LongName='Patient Class' Type='IS' -->
  <PV1.3>
    <PL.1>
      <CM_PL_PRO.1>AEPATZ</CM_PL_PRO.1>
      <!-- Item='4021' Table='0' LongName='' Type='ST' -->
      <CM_PL_PRO.2>Ermächtigung CA Dr. Patzer</CM_PL_PRO.2>
      <!-- Item='4022' Table='0' LongName='' Type='ST' -->
      <CM_PL_PRO.3>20030826115500</CM_PL_PRO.3>
      <!-- Item='4023' Table='0' LongName='' Type='ST' -->
    </PL.1>
    <PL.4>
      <CM_PL_PRO.1>KCH</CM_PL_PRO.1>
      <!-- Item='4021' Table='0' LongName='' Type='ST' -->
      <CM_PL_PRO.2>Kinderchirurgie</CM_PL_PRO.2>
      <!-- Item='4022' Table='0' LongName='' Type='ST' -->
      <CM_PL_PRO.3>20030826115500</CM_PL_PRO.3>
      <!-- Item='4023' Table='0' LongName='' Type='ST' -->
    </PL.4>
  </PV1.3>
  <PV1.16>0</PV1.16>
  <!-- Item='146' Table='99' LongName='VIP Indicator' Type='IS' -->
  <PV1.19>
    <CX_PRO.1>287101</CX_PRO.1>
    <!-- LongName='ID' Type='ST' Table='0' -->
    <CX_PRO.4>
      <HD.2>PROSIGHT</HD.2>
      <!-- LongName='universal ID' Type='ST' Table='0' -->
      <HD.3>L</HD.3>
      <!-- LongName='universal ID type' Type='ID' Table='301' -->
    </CX_PRO.4>
    <CX_PRO.5>
      <HD.2>PROSIGHT</HD.2>

```

```

    <!-- LongName='universal ID' Type='ST' Table='0' -->
    <HD.3>L</HD.3>
    <!-- LongName='universal ID type' Type='ID' Table='301' -->
  </CX_PRO.5>
  <CX_PRO.7>
    <TS.1>20030826115500</TS.1>
    <!-- LongName='date/time' Type='ST' Table='0' -->
    <TS.2>YYYYMMDDHHMMSS</TS.2>
    <!-- LongName='degree of precision' Type='ST' Table='0' -->
  </CX_PRO.7>
</PV1.19>
<PV1.25>
  <TS.1>20030826115500</TS.1>
  <!-- LongName='date/time' Type='ST' Table='0' -->
  <TS.2>YYYYMMDDHHMMSS</TS.2>
  <!-- LongName='degree of precision' Type='ST' Table='0' -->
</PV1.25>
<PV1.41>D</PV1.41>
<!-- Item='171' Table='117' LongName='Account Status' Type='IS' -->
<PV1.44>
  <TS.1>20030826115500</TS.1>
  <!-- LongName='date/time' Type='ST' Table='0' -->
  <TS.2>YYYYMMDDHHMMSS</TS.2>
  <!-- LongName='degree of precision' Type='ST' Table='0' -->
</PV1.44>
<PV1.50>
  <CX_PRO.1>186414</CX_PRO.1>
  <!-- LongName='ID' Type='ST' Table='0' -->
  <CX_PRO.3>
    <HD.2>PROSIGHT</HD.2>
    <!-- LongName='universal ID' Type='ST' Table='0' -->
    <HD.3>L</HD.3>
    <!-- LongName='universal ID type' Type='ID' Table='301' -->
  </CX_PRO.3>
  <CX_PRO.4>
    <HD.2>PROSIGHT</HD.2>
    <!-- LongName='universal ID' Type='ST' Table='0' -->
    <HD.3>L</HD.3>
    <!-- LongName='universal ID type' Type='ID' Table='301' -->
  </CX_PRO.4>
  <CX_PRO.6>
    <HD.1>20030826115500</HD.1>
    <!-- LongName='namespace ID' Type='IS' Table='300' -->
    <HD.2>YYYYMMDDHHMMSS</HD.2>
    <!-- LongName='universal ID' Type='ST' Table='0' -->
  </CX_PRO.6>
</PV1.50>
</PV1>
<OBX>
  <OBX.3>
    <CE.1>8345-1</CE.1>
    <!-- LongName='identifier (ID)' Type='ID' Table='0' -->
    <CE.2>Koerpergewicht</CE.2>
    <!-- LongName='text' Type='ST' Table='0' -->
    <CE.3>LN</CE.3>
    <!-- LongName='name of coding system' Type='IS' Table='396' -->
  </OBX.3>
  <OBX.5>0</OBX.5>
  <!-- Item='573' Table='0' LongName='Observation Value' Type='varies' -->
  <OBX.6>
    <CE.1>g</CE.1>
    <!-- LongName='identifier (ID)' Type='ID' Table='0' -->

```

```

        <CE.3>ISO+</CE.3>
        <!-- LongName='name of coding system' Type='IS' Table='396' -->
    </OBX.6>
    <OBX.14>
        <TS.1>200308260000</TS.1>
        <!-- LongName='date/time' Type='ST' Table='0' -->
        <TS.2>YYYYMMDDHHMM</TS.2>
        <!-- LongName='degree of precision' Type='ST' Table='0' -->
    </OBX.14>
</OBX>
<ZBE>
    <ZBE.1>
        <EI.1>1</EI.1>
        <!-- LongName='entity identifier' Type='ST' Table='0' -->
    </ZBE.1>
    <ZBE.2>
        <TS.1>20030826115500</TS.1>
        <!-- LongName='date/time' Type='ST' Table='0' -->
    </ZBE.2>
    <ZBE.4>INSERT</ZBE.4>
    <!-- Item='49074' Table='0' LongName='Grund/Verarbeitungskennzeichen' Type='ST' -->
</ZBE>
</ADT_A01>

```

Das Stylesheet für die Verarbeitung soll die Daten so aufbereiten, dass die Patientendaten in den Ordner Patient, Aufenthaltsdaten in das Register Aufenthalt und Bewegungsdaten in das Dokument Bewegung. Entsprechend der HL7-Spezifikation sollen folgende HL7-Ereignisse verarbeitet werden, A01, A02, A03, A08, A11, A13. Alle anderen Ereignisse werden nicht verarbeitet.

Dafür kann folgendes Stylesheet verwendet werden.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
#####
#####
#
# Importkonfiguration Halle
#
# Inputformat: HL7 (v2.xml)
#
# OS, J. Schuster, 26.08.2003
# Änderungen:
#
#                                     1. 11.09.2003 format-number aus Fall-ID nud Pat-ID
entfernt, da keine führenden Nullen entstehen dürfen
#
#####
#####
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc">
    <xsl:output method="xml" encoding="ISO-8859-1" omit-xml-declaration="yes" indent="yes"/>
    <!--
#####
#####
#      Javascript
#####
#####
-->

```



```
<msxsl:script language="JavaScript" implements-prefix="osc"><![CDATA[
```

```
function formatDate(d)
{
    var ret = "";
    if ( d.length <= 0 ) { return ( ret ); }

    ret += d.substr(6,2);
    ret += ".";
    ret += d.substr(4,2);
    ret += ".";
    ret += d.substr(0,4);

    return (ret );
}
```

```
function formatTime(t)
{
    var ret = t;
    if (ret == "")
        return (ret);
    ret = t.substr(8,2);
    ret += ".";
    ret += t.substr(10,2);
    return (ret);
}
```

```
]]></msxsl:script>
```

```
<!--
```

```
#####
#####
```

```
#
```

```
# Root-Dispatcher
```

```
#
```

```
# Hier wird der HL7-Nachrichtentyp verwendet, um die Struktur und das zu verwendende Template
# zu bestimmen.
```

```
# Zu beachten ist, dass verschiedene Eventtypen (z.B. A01, A08) die gleiche
```

```
# Datenstruktur (z.B. ADT_A01) verwenden.
```

```
#
```

```
# Es wird dann erneut das root-element selektiert und mit dem entsprechenden "mode" aufgerufen
```

```
#
```

```
#####
#####
```

```
-->
```

```
<xsl:template match="/">
```

```
<xsl:variable name="Msg_Type" select="*/MSH/Mvgl.9/CM_MSG.2"/>
```

```
<xsl:text disable-output-escaping="yes">&lt;?xml version="1.0" encoding="ISO-8859-1" ?&gt;</xsl:text>
```

```
<asimport>
```

```
<xsl:choose>
```

```

<xsl:when test="$Msg_Type='A01'">
  <xsl:apply-templates select="/*" mode="A01"/>
</xsl:when>
<xsl:when test="$Msg_Type='A02'">
  <xsl:apply-templates select="/*" mode="A02"/>
</xsl:when>
<xsl:when test="$Msg_Type='A03'">
  <xsl:apply-templates select="/*" mode="A03"/>
</xsl:when>
<xsl:when test="$Msg_Type='A08'">
  <xsl:apply-templates select="/*" mode="A08"/>
</xsl:when>
<xsl:when test="$Msg_Type='A11'">
  <xsl:apply-templates select="/*" mode="A11"/>
</xsl:when>
<xsl:when test="$Msg_Type='A13'">
  <xsl:apply-templates select="/*" mode="A13"/>
</xsl:when>
<xsl:otherwise>
  <xsl:text>Diese Nachricht wird nicht verarbeitet! (</xsl:text>
  <xsl:value-of select="$Msg_Type"/>
  <xsl:text>)</xsl:text>
</xsl:otherwise>
</xsl:choose>
<!-- Einfügen des Original-XML zu Testzwecken -->
<!--
<orig>
  <xsl:copy-of select="ADT_A01/*"/>
</orig>
-->
</asimport>
</xsl:template>
<!--
#####
#####
#
# A01 - Aufnahme
#
#####
#####
-->
<xsl:template match="/*" mode="A01">
  <xsl:comment>*** AUFNAHME ***</xsl:comment>
  <!-- Task Stammdaten -->
  <task>
    <ord name="Patient" dbname="stamm1" cotype="4">
      <data>
        <xsl:call-template name="Patient-Ordner">
          <xsl:with-param name="aktion">select-or-
insert</xsl:with-param>

```

```

        </xsl:call-template>
    </data>
    <reg name="AUFENTHALT" dbname="register1" cotype="0">
        <data>
            <xsl:call-template name="Fall-Register">
                <xsl:with-param
name="storno">nein</xsl:with-param>
            </xsl:call-template>
        </data>
        <doc name="Bewegung" maintype="4">
            <data>
                <xsl:call-template name="Bewegung-
INSERT"/>
            </data>
        </doc>
    </reg>
</ord>
</task>
</xsl:template>
<!--
#####
#####
#
# A02 - Verlegung
#
#####
#####
-->
<xsl:template match="/*" mode="A02">
    <xsl:comment>*** VERLEGUNG ***</xsl:comment>
    <!-- Task Stammdaten -->
    <task>
        <ord name="Patient" dbname="stamm1" cotype="4">
            <data>
                <xsl:call-template name="Patient-Ordner">
                    <xsl:with-param name="aktion">select-or-
insert</xsl:with-param>
                </xsl:call-template>
            </data>
            <reg name="AUFENTHALT" dbname="register1" cotype="0">
                <data>
                    <xsl:call-template name="Fall-Register">
                        <xsl:with-param
name="storno">nein</xsl:with-param>
                    </xsl:call-template>
                </data>
                <doc name="Bewegung" maintype="4">
                    <data>
                        <xsl:call-template name="Bewegung-
INSERT"/>
                    </data>
                </doc>
            </reg>
        </ord>
    </task>
</xsl:template>

```

```

</data>
</doc>
</reg>
</ord>
</task>
</xsl:template>
<!--
#####
#####
#
# A03 - Entlassung
#
#####
#####
-->
<xsl:template match="/" mode="A03">
  <xsl:comment>*** ENTLASSUNG ***</xsl:comment>
  <!-- Task Stammdaten -->
  <task>
    <ord name="Patient" dbname="stamm1" cotype="4">
      <data>
        <xsl:call-template name="Patient-Ordner">
          <xsl:with-param name="aktion">select-or-
insert</xsl:with-param>
          </xsl:call-template>
        </data>
        <reg name="AUFENTHALT" dbname="register1" cotype="0">
          <data>
            <xsl:call-template name="Fall-Register">
              <xsl:with-param
name="storno">nein</xsl:with-param>
            </xsl:call-template>
          </data>
          <doc name="Bewegung" maintype="4">
            <data>
              <xsl:call-template name="Bewegung-
INSERT"/>
            </data>
          </doc>
        </reg>
      </ord>
    </task>
  </xsl:template>
<!--
#####
#####
#
# A08 - Update
#
#####
#####

```

```
#####
-->
<xsl:template match="/*" mode="A08">
  <xsl:comment>*** UPDATE ***</xsl:comment>
  <!-- Task Stammdaten -->
  <task>
    <ord name="Patient" dbname="stamm1" cotype="4">
      <data>
        <xsl:call-template name="Patient-Ordner">
          <xsl:with-param name="aktion">select-or-
insert</xsl:with-param>
          </xsl:call-template>
        </data>
        <reg name="AUFENTHALT" dbname="register1" cotype="0">
          <data>
            <xsl:call-template name="Fall-Register">
              <xsl:with-param
name="storno">nein</xsl:with-param>
              </xsl:call-template>
            </data>
            <doc name="Bewegung" maintype="4">
              <data>
                <xsl:call-template name="Bewegung-
UPDATE"/>
              </data>
            </doc>
          </reg>
        </ord>
      </task>
    </xsl:template>
  <!--
#####
#####
#
# A11 Storno Aufnahme
#
#####
#####
-->
<xsl:template match="/*" mode="A11">
  <xsl:comment>*** STORNO Aufnahme ***</xsl:comment>
  <!-- !! TODO !!
  <task>
    <xsl:call-template name="Patient-Ordner"/>
  </task>
  -->
  <task>
    <ord name="Patient" dbname="stamm1" cotype="4">
      <data>
        <xsl:call-template name="Patient-Ordner">
```

```

                                <xsl:with-param name="aktion">select</xsl:with-
param>
                                </xsl:call-template>
                                </data>
                                <reg name="AUFENTHALT" dbname="register1" cotype="0">
                                <data>
                                <xsl:call-template name="Fall-Register">
                                <xsl:with-param
name="storno">Aufenthalt</xsl:with-param>
                                </xsl:call-template>
                                </data>
                                <doc name="Bewegung" maintype="4">
                                <data>
                                <xsl:call-template name="Bewegung-
DELETE"/>
                                </data>
                                </doc>
                                </reg>
                                </ord>
                                </task>
                                </xsl:template>
                                <!--
#####
#####
#
# A13 - Entlassung
#
#####
#####
-->
                                <xsl:template match="/*" mode="A13">
                                <xsl:comment>*** STORNO Entlassung ***</xsl:comment>
                                <!-- !! TODO !!
                                <task>
                                <xsl:call-template name="Patient-Ordner"/>
                                </task>
                                -->
                                <task>
                                <ord name="Patient" dbname="stamm1" cotype="4">
                                <data>
                                <xsl:call-template name="Patient-Ordner">
                                <xsl:with-param name="aktion">select</xsl:with-
param>
                                </xsl:call-template>
                                </data>
                                <reg name="AUFENTHALT" dbname="register1" cotype="0">
                                <data>
                                <xsl:call-template name="Fall-Register">
                                <xsl:with-param
name="storno">Entlassung</xsl:with-param>

```

```

        </xsl:call-template>
    </data>
    <doc name="Bewegung" maintype="4">
        <data>
            <xsl:call-template name="Bewegung-
DELETE"/>
        </data>
    </doc>
</reg>
</ord>
</task>
</xsl:template>

<!--
#####
#####
#
# Template für Patientenstammdaten (OS:EPA Ordner)
#
#####
#####
-->
    <xsl:template name="Patient-Ordner">
        <xsl:param name="aktion"/>
        <!--
        #
        # Variablen für spezielle Segmente
        #
        -->
        <!--
        #
        # Formatierung und Mapping
        #
        -->
        <!--
        # variable
        -->
<!-- siehe 1.Änderung -->
    <xsl:variable name="PAT_ID" select="PID/PID.3/CX.1"/>

<!--
    <xsl:variable name="PAT_ID">
        <xsl:variable name="VAL" select="PID/PID.3/CX.1"/>
        <xsl:choose>
            <xsl:when test="number($VAL)">
                <xsl:value-of select="format-number($VAL,'000000000')"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="$VAL"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:variable>

```

```

</xsl:variable> -->
<!--
# variable
-->
<xsl:variable name="Geschlecht-mapped">
  <xsl:variable name="VAL" select="PID/PID.8"/>
  <xsl:choose>
    <xsl:when test="$VAL='M'">männlich</xsl:when>
    <xsl:when test="$VAL='W' or $VAL='F'">weiblich</xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$VAL"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
<!--
#
# Import-Anweisungen
#
-->
<xsl:choose>
  <xsl:when test="$aktion != 'select'">
    <select-or-insert name="Patienten-ID" value="{\$PAT_ID}"/>
    <update name="Name" value="{normalize-
space(concat(PID/PID.5[1]/XPN.1/FN.1,' ',PID/PID.5[1]/XPN.1/FN.2))}"/>
    <update name="Vorname" value="{PID/PID.5[1]/XPN.2}"/>
    <update name="Titel" value="{PID/PID.5[1]/XPN.6}"/>
    <update name="Geschlecht" value="{PID/PID.8}"/>
    <update name="Geburtsdatum"
value="{osc:formatDate(string(PID/PID.7/TS.1))}"/>
    <update name="Geburtsname" value="{PID/PID.6/XPN.1}"/>
    <update name="Straße / Hausnr."
value="{PID/PID.11/XAD.1/SAD.1[../XAD.7 = 'H']}"/>
    <update name="PLZ" value="{PID/PID.11/XAD.5[../XAD.7 = 'H']}"/>
    <update name="Stadt" value="{PID/PID.11/XAD.3[../XAD.7 = 'H']}"/>
    <update name="Telefon" value="{PID/PID.13/XTN.1}"/>
  </xsl:when>
  <xsl:otherwise>
    <select name="Patienten-ID" value="{\$PAT_ID}"/>
  </xsl:otherwise>
</xsl:choose>
</xsl:template>
<!--
#####
#####
#
# Template für Aufenthaltsdatendaten (OS:EPA Register)
#
#####
#####
-->

```



```

<!--[PATIENT].[Aufenthalt]-->
<xsl:template name="Fall-Register">
  <xsl:param name="storno" select="nein"/>
  <!--
  #
  # Variablen für spezielle Segmente
  #
  -->

  <!--
  #
  # Formatierung und Mapping
  #
  -->
  <!--
  # variable
  -->

<!-- siehe 1.Änderung -->
  <xsl:variable name="FALL_ID" select="PV1/PV1.19/CX_PRO.1"/>

<!--
  <xsl:variable name="FALL_ID">
    <xsl:variable name="VAL" select="PV1/PV1.19/CX_PRO.1"/>
    <xsl:choose>
      <xsl:when test="number($VAL)">
        <xsl:value-of select="format-number($VAL,'00000000')"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$VAL"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable> -->
  <!--
  # variable
  -->
  <xsl:variable name="FACHABTEILUNG">
<!--
    <xsl:variable name="VAL" select="PV1/PV1.3/PL.4/HD.1"/> -->
    <xsl:variable name="VAL" select="PV1/PV1.3/PL.7"/>
    <xsl:choose>
      <xsl:when test="number($VAL)">
<!--
        <xsl:value-of select="format-number($VAL,'###')"/> -->
        <xsl:value-of select="$VAL"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$VAL"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <!--
  #

```

```

# Import-Anweisungen
#
-->
<xsl:choose>
  <xsl:when test="$storno = 'nein'">
    <select-or-insert name="Fall-ID" value="{FALL_ID}"/>
<!-- achte auf PV1.19 für Aufnahme und Entlassung -->
<!--
      <update name="Aufnahmedatum"
value="{osc:formatDate(string(PV1/PV1.44/TS.1))}"/>
      <update name="Aufnahmezeit"
value="{osc:formatTime(string(PV1/PV1.44/TS.1))}"/> -->
      <update name="Aufnahmedatum"
value="{osc:formatDate(string(PV1/PV1.19/CX_PRO.7/TS.1))}"/>
      <update name="Aufnahmezeit"
value="{osc:formatTime(string(PV1/PV1.19/CX_PRO.7/TS.1))}"/>
      <update name="Aufnahmeart" value="{PV1/PV1.4}"/>
<!--
      <update name="Entlassungsdatum"
value="{osc:formatDate(string(PV1/PV1.45/TS.1))}"/>
      <update name="Entlassungszeit"
value="{osc:formatTime(string(PV1/PV1.45/TS.1))}"/> -->
      <update name="Entlassungsdatum"
value="{osc:formatDate(string(PV1/PV1.19/CX_PRO.8/TS.1))}"/>
      <update name="Entlassungszeit"
value="{osc:formatTime(string(PV1/PV1.19/CX_PRO.8/TS.1))}"/>
      <update name="Entlassungsart" value="{PV1/PV1.36}"/>
      <update name="Station/OE"
value="{PV1/PV1.3/PL.1/CM_PL_PRO.1}"/>
<!--
      <update name="Fachabteilung" value="{FACHABTEILUNG}"/>
-->

      <update name="Patientenstatus" value="{PV1/PV1.2}"/>
    </xsl:when>
    <xsl:when test="$storno = 'Entlassung'">
      <select name="Fall-ID" value="{FALL_ID}"/>
      <update name="Entlassungsdatum" value="empty"/>
      <update name="Entlassungszeit" value="empty"/>
      <update name="Entlassungsart" value="empty"/>
    </xsl:when>
    <xsl:when test="$storno = 'Aufenthalt'">
      <select name="Fall-ID" value="{FALL_ID}"/>
      <update name="Aufenthalt storniert" value="1"/>
    </xsl:when>
  </xsl:choose>
</xsl:template>
<!--
#####
#####
#
# Template für das Anlegen von Bewegungen (OS:EPA Dokument)
#
#####
#####

```

```

-->
<xsl:template name="Bewegung-INSERT">
    <select-or-insert name="Zeitpunkt" value="{EVN/EVN.2/TS.1}"/>
    <update name="Bewegungsart" value="{EVN/EVN.1}"/>
    <update name="Patientenstatus" value="{PV1/PV1.2}"/>
    <update name="KSt" value="{PV1/PV1.39}"/>
<!--
    <update name="Fachabteilung" value="{number(PV1/PV1.3/PL.4/HD.1)}"/> -->
    <update name="Fachabteilung" value="{PV1/PV1.3/PL.7}"/>
    <update name="Station" value="{PV1/PV1.3/PL.1/CM_PL_PRO.1}"/>
</xsl:template>
<!--
#####
#####
#
# Template für das Ändern von Bewegungen (OS:EPA Dokument)
#
#####
#####
-->
<xsl:template name="Bewegung-UPDATE">
    <select-or-insert name="Zeitpunkt" value="{EVN/EVN.2/TS.1}"/>
    <!-- update name="Bewegungsart" value="{EVN/EVN.1}"/ -->
    <!-- nicht ändern -->
    <update name="Patientenstatus" value="{PV1/PV1.2}"/>
    <update name="KSt" value="{PV1/PV1.39}"/>
<!--
    <update name="Fachabteilung" value="{number(PV1/PV1.3/PL.4/HD.1)}"/> -->
    <update name="Fachabteilung" value="{PV1/PV1.3/PL.7}"/>
    <update name="Station" value="{PV1/PV1.3/PL.1/CM_PL_PRO.1}"/>
</xsl:template>
<!--
#####
#####
#
# Template für das Stornieren von Bewegungen (OS:EPA Dokument)
#
#####
#####
-->
<xsl:template name="Bewegung-DELETE">
    <select name="Zeitpunkt" value="{EVN/EVN.2/TS.1}"/>
    <update name="Bewegungsart" value="{EVN/EVN.1}"/>
    <update name="Patientenstatus" value="{PV1/PV1.2}"/>
    <update name="KSt" value="{PV1/PV1.39}"/>
<!--
    <update name="Fachabteilung" value="{number(PV1/PV1.3/PL.4/HD.1)}"/> -->
    <update name="Fachabteilung" value="{PV1/PV1.3/PL.7}"/>
    <update name="Station" value="{PV1/PV1.3/PL.1/CM_PL_PRO.1}"/>
</xsl:template>
<!--
#####
#####

```

```

#
# Default Template
#
#####
#####
-->
    <!-- default: suppress unmatched nodes -->
    <xsl:template match="node()|@*" />
</xsl:stylesheet>
Das Ergebnis der Transformation ist wie folgt. Dieses Format kann anschließend von der Komponente
OscConImpE.Exe verarbeitet werden.
<?xml version="1.0" encoding="ISO-8859-1"?>
<asimport xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:osc="urn:optimal-systems-de:osc">
    <!-- *** AUFNAHME *** -->
    <task>
        <ord name="Patient" dbname="stamm1" cotype="4">
            <data>
                <select-or-insert name="Patienten-ID" value="000140603"/>
                <update name="Name" value="Mustermann"/>
                <update name="Vorname" value="Manfred"/>
                <update name="Titel" value=""/>
                <update name="Geschlecht" value="M"/>
                <update name="Geburtsdatum" value="19.02.1987"/>
                <update name="Geburtsname" value=""/>
                <update name="Straße / Hausnr." value="Mustertr. 33"/>
                <update name="PLZ" value="06682"/>
                <update name="Stadt" value="Teuchern"/>
                <update name="Telefon" value=""/>
            </data>
            <reg name="AUFENTHALT" dbname="register1" cotype="0">
                <data>
                    <select-or-insert name="Fall-ID" value=""/>
                    <update name="Aufnahmedatum" value="26.08.2003"/>
                    <update name="Aufnahmezeit" value="11:55"/>
                    <update name="Aufnahmeart" value=""/>
                    <update name="Entlassungsdatum" value=""/>
                    <update name="Entlassungszeit" value=""/>
                    <update name="Entlassungsart" value=""/>
                    <update name="Station/OE" value="AEPATZ Ermächtigung
CA Dr. Patzer 20030826115500"/>
                    <update name="Fachabteilung" value=""/>
                    <update name="Patientenstatus" value="O"/>
                </data>
                <doc name="Bewegung" maintype="4">
                    <data>
                        <select-or-insert name="Zeitpunkt"
value="20030826115535"/>
                        <update name="Bewegungsart" value="A01"/>
                        <update name="Patientenstatus" value="O"/>
                        <update name="KSt" value=""/>
                    </data>
                </doc>
            </reg>
        </ord>
    </task>
</asimport>

```

```

        <update name="Fachabteilung" value=""/>
        <update name="Station" value="AEPATZ
Ermächtigung CA Dr. Patzer 20030826115500"/>
    </data>
</doc>
</reg>
</ord>
</task>
</asimport>

```

OscPassThrough

Einleitung

Die Komponente OscPassThrough (OscPassThroughTransformer.Dll) ist ein Transformer der Daten unmanipuliert weiterreicht. Diese Komponente kann eingesetzt werden, um Daten von einem inbound- zu einem outbound-Konnektor unmanipuliert zu übergeben. Für Debug-Zwecke kann diese Komponente ebenfalls eingesetzt werden.

Installation

Die Komponente OscPassThroughTransformer.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. OscConFile ist Bestandteil jeder enaio® communicator Installation.

Konfiguration

OscPassThrough wird nur in der aktiven Konfiguration konfiguriert. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen.

```

<rule name="rl_pass" transformer="Passthrough" definition="">
    <input messagetype="mtimp"/>
    <output messagetype="mtimp"/>
</rule>
<transformation name="tr_pass" rule="rlpass">
    <input supplier="cnin" messagetype="mtimp"/>
    <output messagetype="mtimp"/>
</transformation>

```

OscTrHL7Router

Einleitung

Der Transformer OscTrHL7Router.Dll ist eine ActiveX-Dll, mit deren HL7-Nachrichten gezielt anhand des HL7-Events an Komponenten versendet werden können.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-

		Komponente
Msoxml4.dll	Microsoft	

Installation

Die Komponente OscTrHL7Router.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Standardmäßig steht diese Komponente nur zur Verfügung, wenn die Option enaio® communicator installieren aus dem Setup gewählt wurde. Bei älteren Installationen wird empfohlen, die Komponente OscTrHL7Router.Dll ebenfalls im bin-Verzeichnis abzulegen. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rHL7Router"
  transformer="C"
  definition="OscComTransformer"
  initfile="modules.xml">
  <input messageType="mtHL7XML"/>
  <output messageType="mtADTXML"/>
  <output messageType="mtORUXML"/>
  <output messageType="mtMDMXML"/>
</rule>
<transformation name="trHL7Router" rule="rHL7Router">
  <input supplier="trHL7XML" messageType=" mtHL7XML "/>
  <output messageType="mtADTXML"/>
  <output messageType="mtORUXML"/>
  <output messageType="mtMDMXML"/>
</transformation>
```

Außerdem setzt OscTrHL7Router einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name=" rROUTE ">
  <!-- Feldnamen stehen in der Datei (erste oder zweite Zeile in Abhängigkeit von
filename, Eingabe 0 oder 1 -->
  <topics> A01,A02,A03=trADTXML@trHL7Router;A01,T02=trORUXML@trHL7Router
</topics>
  <!--Path gibt an, wie der Nachrichtentyp ausgelesen werden kann. Fehlt dieser Eintrag, dann
wird der Default-Path verwendet. -->
  <path>*/MSH/MSH.9/MSG.2</path>

  <!-- ProglD der Komponente -->
  <ComProglD>OscTrHL7Router.Trans</ComProglD>
</module>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

topics

Für die output-message-type der Komponenten werden die Ereignisse gesetzt. Es wird folgender Syntax verwendet Event1,Event2,Event*=messageType@Komponente.

Diese Einträge können wiederum mit einem Semikolon getrennt angegeben werden, wenn mehrere output-message-types publiziert werden sollen durch die Komponente´.

Path zum Messagetypp der HL7-Nachricht

Per Default werden folgende Pfade nach dem Messagetypp der HL7-Nachricht abgesucht.

/hl7:*/hl7:MSH/hl7:MSH.9/hl7:CM_MSG_TYPE.2 und /*/MSH/MSH.9/CM_MSG_TYPE.2

Um das Anpassen der HL7.DAT zu vermeiden, kann in der Konfiguration festgelegt werden, welcher Pfad zur Suche verwendet werden soll. Dies geschieht durch den Eintrag **path**.

Beispiel:

```
<path>/*/MSH/MSH.9/MSG.2</path>
```

Encoding-Transformer – OscTrEncoding

Einleitung

Der enaio® communicator arbeitet intern mit einer ACSII-Codierung. UTF-8 und UTF-16-kodierte Daten können damit nicht direkt verarbeitet werden. Um Daten dieser Formate zu verarbeiten, ist eine Transformation mittels Encoding-Transformers notwendig.

Der Encoding-Transformer ermöglicht die Übersetzung der Daten in ein anderes Encoding-Format.

Dieser Transformator sollte vor der Weiterverarbeitung durch andere Transformatoren eingesetzt werden, wenn das Eingangsformat UFT-8 bzw. UFT-16 ist.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. Dort wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="riEncode" transformer="C" definition="OscTrEncoding" initfile="modules.xml">
  <input messagetype="mtIn" />
  <output messagetype="mtOut" />
</rule>
<transformation name="trEncode" rule="riEncode">
  <input supplier="cnIn" messagetype="mtIn" />
  <output messagetype="mtOut" />
</transformation>
```

Außerdem setzt der Transformer einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle möglichen Konfigurationseinstellungen gezeigt:

```
<!--
  Einstellungen fuer den Encode-Transformer
-->
```

```
<module name="riEncode">
```

```
<!--
```

moegliche Werte fuer Source-Encoding:

auto: Es wird versucht, anhand der Dateikennung zu erkennen, ob es sich um eine UTF-Kodierung handelt. Unterstuetzt wird derzeit lediglich UTF-16-LE. Wird in der Datei eine UTF-16-BE-Kennung gefunden, wird ein Fehler geloggt. Alle anderen Dateien werden als ASCII interpretiert.

ASCII: Die Daten werden als ASCII interpretiert.

UTF-16LE:Die Daten werden als UTF-16 little endian interpretiert.

UTF-8 :Die Daten werden als UTF-8 interpretiert

wird dieser Parameter nicht angegeben, wird auto verwendet.

```
-->
```

```
<source-encoding>auto</source-encoding>
```

```
<!--
```

Wenn nicht anhand der BOM ermittelt werden konnte, um welches Encoding es sich handelt ist es möglich, den Transformer anzuweisen, anhand der Kennung im XML-String zu überprüfen, welches Encoding vorliegt. Dies erfolgt nur, wenn der Parameter source-typ den Wert XML besitzt. Wird dieser Parameter nicht angegeben, erfolgt keine Prüfung des enthaltenen Strings.

```
-->
```

```
<source-typ>XML</source-typ>
```

```
<!--
```

moegliche Werte fuer Ziel-Encoding:

ASCII: Die Daten werden in das ASCII-Format kodiert.

UTF-16LE:Die Daten werden in das UTF-16 little endian-Format kodiert.

UTF-8 :Die Daten werden als UTF-8 kodiert.

wird dieser Parameter nicht angegeben, wird ASCII verwendet.

```
-->
```

```
<dest-encoding>ASCII</dest-encoding>
```

```
</module>
```

Bei der Dekodierung geht der Transformer folgendermaßen vor:

Wenn eine Kennung in `<source-encoding>` angegeben wird, wird diese verwendet

Wenn `<source-encoding>` auf **auto** gesetzt, bzw. dieser Parameter nicht angegeben ist:

Es wird nach einem BOM (Byte Order Mark) am Anfang der Daten gesucht. Wird eine Kennung für UTF-8 bzw UTF-16 little endian gefunden, wird dieses Format verwendet. Bei UTF-16 big endian wird ein Fehler gemeldet.

Wird keine Kennung gefunden, werden:

wenn `<source-typ>` auf XML gesetzt ist die Daten auf eine entsprechende Kennung im XML geprüft und diese verwendet. Diese Kennungsprüfung erfolgt derzeit lediglich durch eine Stringsuche nach folgenden „codings“:

utf-8

iso-8859-1

utf-16 (little endian wird angenommen)

Wenn `<source-typ>` auf einen anderen Wert bzw. nicht gesetzt ist, wird angenommen, dass es sich um ASCII-Daten handelt.

CSV-Transformer - OscCsvXml

Einleitung

Der Transformer OscCsvXml.dll ist eine ActiveX-Dll, mit deren Hilfe Csv-Daten(komma-separiert) in ein einheitliches Xml-Format (vgl. Ausgangsformat von OscCsvXml) transformiert werden können. Durch dieses einheitliche Format, kann dann sehr bequem aus einem Stylesheet oder einem anderen Programm auf diese Daten zugegriffen werden. Pro Zeile aus dem Quelldokument wird ein Datensatz generiert.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente
Axvbxml.dll	enaio®	enaio® communicator-Komponente
Msoxml4.dll	Microsoft	

Installation

Die Komponente OscCsvXml.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein. Diese Komponente wird im bin-Verzeichnis des enaio® communicator abgelegt.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rl_Csv_Xml"
  transformer="C"
  definition="OscComTransformer"
  initfile="modules.xml">
  <input messagetype="mtCsv"/>
  <output messagetype="mtCsvXml"/>
</rule>
<transformation name="tr_Csv_Xml" rule="rl_Csv_Xml">
  <input supplier="cn_Csv_In" messagetype="mtCsv"/>
  <output messagetype="mtCsvXml"/>
</transformation>
```

Außerdem setzt OscCsvXml einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="rl_Csv_Xml">
```

```
  <!-- Feldnamen stehen in der Datei (erste oder zweite Zeile in Abhängigkeit von
filename, Eingabe 0 oder 1 -->
```

```
  <fieldnames>>false</fieldnames>
```

```
  <!-- In der ersten Zeile wird der Dateiname übergeben, Eingabe 0 oder 1 -->
```

```
  <filename>>false</filename>
```

```
  <!-- Feldtrennzeichen -->
```

```
  <fieldsep>;</fieldsep>
```

```
  <!-- Satztrennzeichen, werden mit Semikolon getrennt dargestellt -->
```

```
  <recsep>13;10</recsep>
```

```
  <!-- Textzeichen, bedeutet das innerhalb dieses Textes das Feldtrennzeichen
vorkommen darf -->
```

```
  <txtsign></txtsign>
```

```

<!-- Zeile, ab der verarbeitet werden soll -->
<begin>1</begin>
<!--Sollen die Daten zeilenweise verarbeitet werden -->
<line-by-line>0</line-by-line>
<!-- encoding Deklaration, ISO_8859-1, UTF-8, UTF-16 -->
<encoding>ISO-8859-1</encoding>
<!--Ein Csv-Feld besteht aus einem Paar name=wert -->
<fieldiskeyvaluepair/>
<!-- ProgId der komponente -->
<ComProgId>OscCsvXml.Trans</ComProgId>
</module>

```

Als Mindestkonfiguration werden folgende Angaben benötigt:

recsep
fieldsep
fieldnames
filename
line-by-line
ComProgId

Die `ComProgId` muss angegeben werden. Ab dann wird das Attribut `recsep` auf `carriage return line feed` als Standard gesetzt. Das Attribut `fieldsep` wird dann als Standard auf Semikolon gesetzt. Die Attribute `fieldnames` und `filename` werden jeweils als `false` gesetzt.

Ist das Attribut `fieldnames` gesetzt, werden die Werte aus der ersten bzw. zweiten Zeile (abhängig von dem Status von `filename`) der übergebenen Csv-Daten als Feldnamen angenommen. Ist `filename` gesetzt, kann der Dateiname in der ersten Zeile übergeben werden.

Mit dem Attribut `fieldsep` wird der Separator der einzelnen Felder bestimmt. Standardmäßig wird hier das Semikolon gesetzt. Mit `recsep` wird bestimmt, welches Zeichen der Datensatztrenner ist. Standardmäßig wird „13;10“ (`carriage return line feed`) gesetzt.

Mit `txtsign` kann bestimmt werden, dass innerhalb eines Feldes ein Text beispielsweise in Anführungszeichen steht und in diesem Bereich auch Trennzeichen vorkommen können.

Mit `begin` wird die Zeile angegeben, ab der die Daten verarbeitet werden sollen. Meistens sollte das die erste Zeile sein.

Mit `encoding` kann die Xml-Deklaration für den Output gesteuert werden. Standardmäßig wird hier „ISO-8859-1“ gesetzt.

Mit `line-by-line` kann bestimmt werden, dass die Csv-Daten zeilenweise verarbeitet werden. Standardmäßig wird hier „0“ gesetzt.

Mit dem Attribut `fieldiskeyvaluepair` kann gesteuert werden, dass ein als Trennzeichen als Untertrennzeichen innerhalb eines Feldes angewendet wird. Es kann jedes beliebige Zeichen ungleich `fieldsep` und `recsep` verwendet werden.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Die weitere Verarbeitung wird meistens mit XSLT-Transformationen innerhalb des enaio® communicator realisiert.

Ein zusätzlicher Konfigurationsparameter `trim-values` (Standard = `true`) wird unterstützt. Wird `trim-values` auf `false` gesetzt werden Leerzeichen nicht mehr aus den Wertwen getrimmt.

Ausgangsformat von OscCsvXml

Aus dem folgenden Schema kann der Aufbau des generierten Xml-Formates des Transformiers entnommen werden. Das Format von `OscCsvXml` entspricht dem der Komponente `OscFixXml.dll`. Einziger Unterschied ist, dass bei `OscCsvXml` im Root-Tag zusätzlich das Attribut `sourcefile` angegeben werden kann.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="osc">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="row" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="data-format" type="xs:string" use="optional"/>
      <xs:attribute name="sourcefile" type="xs:string" use="optional"/>
      <xs:attribute name="processing" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="row">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="position" type="xs:int" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="item">
    <xs:complexType>
      <xs:attribute name="key" type="xs:string" use="required"/>
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Die schematische Darstellung (Bild 14):

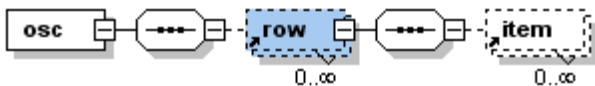


Abbildung 21

Aufbau Csv-Xml

Ein Beispieldatensatz für Csv-Xml:

```

<?xml version="1.0"?>
<osc data-format="csv" sourcefile="" processing="ISO-8859-1">
  <row position="1">
    <item key="field0" value="Mustermann"/>
    <item key="field1" value="Manfred"/>
    <item key="field2" value="50"/>
    <item key="field3" value="männlich"/>
  </row>
  <row position="2">
    <item key="field0" value="Musterfrau"/>
    <item key="field1" value="Gerda"/>
    <item key="field2" value="50"/>
  </row>
</osc>

```

```

        <item key="field3" value="weiblich"/>
    </row>
</osc>

```

Beispiele für Konfigurationen und ihre Transformationsergebnisse

Standard-Csv

Konfiguration:

```

<module name="osccsvxml">
    <recsep>13;10</recsep>
    <fieldsep>;</fieldsep>
    <fieldnames>>false</fieldnames>
    <filename>>false</filename>
    <line-by-line>0</line-by-line>
    <encoding>ISO-8859-1</encoding>
    <ComProgId>OscCsvXml.Trans</ComProgId>
</module>

```

Dateninput:

Mustermann;Manfred;50;männlich

Musterfrau;Gerda;50;weiblich

Ergebnis:

```

<?xml version="1.0"?>
<osc data-format="csv" sourcefile="" processing="ISO-8859-1">
    <row position="1">
        <item key="field0" value="Mustermann"/>
        <item key="field1" value="Manfred"/>
        <item key="field2" value="50"/>
        <item key="field3" value="männlich"/>
    </row>
    <row position="2">
        <item key="field0" value="Musterfrau"/>
        <item key="field1" value="Gerda"/>
        <item key="field2" value="50"/>
        <item key="field3" value="weiblich"/>
    </row>
</osc>

```

Csv mit Feld- und Dateiheader

Konfiguration:

```

<module name="osccsvxml">
    <recsep>13;10</recsep>
    <fieldsep>;</fieldsep>
    <fieldnames>>true</fieldnames>
    <filename>>true</filename>
    <line-by-line>0</line-by-line>
    <encoding>ISO-8859-1</encoding>
    <ComProgId>OscCsvXml.Trans</ComProgId>
</module>

```

Dateninput:

003.csv

Nachname;Vorname;Alter;Geschlecht

Mustermann;Manfred;50;männlich

Musterfrau;Gerda;50;weiblich

Ergebnis:

```
<?xml version="1.0"?>
<osc data-format="csv" sourcefile="003.csv" processing="ISO-8859-1">
  <row position="1">
    <item key="Nachname" value="Mustermann"/>
    <item key="Vorname" value="Manfred"/>
    <item key="Alter" value="50"/>
    <item key="Geschlecht" value="männlich"/>
  </row>
  <row position="2">
    <item key="Nachname" value="Musterfrau"/>
    <item key="Vorname" value="Gerda"/>
    <item key="Alter" value="50"/>
    <item key="Geschlecht" value="weiblich"/>
  </row>
</osc>
```

Wird nur eine Option, entweder `filename` oder `fieldnames` angewendet, wird entsprechend dem Eintrag in der ersten Zeile der Daten der Output generiert.

Felder mit Textzeichen

Konfiguration:

```
<module name="osccsvxml">
  <recsep>13;10</recsep>
  <fieldsep>;</fieldsep>
  <fieldnames>>false</fieldnames>
  <filename>>false</filename>
  <line-by-line>0</line-by-line>
  <encoding>ISO-8859-1</encoding>
  <txtsign>"</txtsign>
  <ComProgId>OscCsvXml.Trans</ComProgId>
</module>
```

Dateninput:

Mustermann;Manfred;50;männlich";"weiblich

Musterfrau;Gerda;50;weiblich";"männlich

Ergebnis:

```
<?xml version="1.0"?>
<osc data-format="csv" sourcefile="" processing="ISO-8859-1">
  <row position="1">
    <item key="field0" value="Mustermann"/>
    <item key="field1" value="Manfred"/>
    <item key="field2" value="50"/>
    <item key="field3" value="männlich;weiblich"/>
  </row>
  <row position="2">
```

```

    <item key="field0" value="Musterfrau"/>
    <item key="field1" value="Gerda"/>
    <item key="field2" value="50"/>
    <item key="field3" value="weiblich;männlich"/>
  </row>
</osc>

```

Felder mit Name-Wert-Paar

Bei diesem Beispiel wird mit Hilfe von OscCsvXml ein Datensatz verarbeitet, in dem pro Zeile jeweils ein Name-Wert-Paar abgelegt wurde. Mit Hilfe der Einstellung `fieldiskeyvaluepair=""` wird erreicht, dass das gesamte Feld (in diesem Fall eine Zeile) nochmal getrennt werden kann.

Konfiguration:

```

<module name="osccsvxml">
  <fieldiskeyvaluepair=</fieldiskeyvaluepair>
  <fieldnames>false</fieldnames>
  <filename>false</filename>
  <fieldsep>10</fieldsep>
  <recsep>13;10;13;10</recsep>
  <txtsign>"</txtsign>
  <line-by-line>0</line-by-line>
  <begin>1</begin>
  <encoding>ISO-8859-1</encoding>
  <ComProgId>OscCsvXml.Trans</ComProgId>
</module>

```

Dateninput:

```

(0008,0020)=20000101
(0008,0030)=145050.00000
(0008,0050)=
(0008,0060)=CT
(0008,0090)=Dr. Nobody
(0010,0010)=Anonymized
(0010,0020)=0123456789
(0010,0030)=20000101
(0010,0040)=M
(0020,0010)=2019
(0020,0011)=1
(0020,000D)=1.3.46.670589.10.900123.19970114.35056000060.1
(0020,000E)=1.3.46.670589.10.900123.19970114.35056000060.3
(9999,9995)=
(9999,9997)=5
(9999,9999)=\ser1055846208920

```

Ergebnis:

```

<?xml version="1.0"?>
<osc data-format="csv" sourcefile="" processing="ISO-8859-1">
  <row position="1">
    <item key="(0008,0020)" value="20000101"/>
    <item key="(0008,0030)" value="145050.00000"/>
    <item key="(0008,0050)" value=""/>
    <item key="(0008,0060)" value="CT"/>
  </row>
</osc>

```

```

<item key="(0008,0090)" value="Dr. Nobody"/>
<item key="(0010,0010)" value="Anonymized"/>
<item key="(0010,0020)" value="0123456789"/>
<item key="(0010,0030)" value="20000101"/>
<item key="(0010,0040)" value="M "/>
<item key="(0020,0010)" value="2019"/>
<item key="(0020,0011)" value="1 "/>
<item key="(0020,000D)"
value="1.3.46.670589.10.900123.19970114.35056000060.1"/>
<item key="(0020,000E)"
value="1.3.46.670589.10.900123.19970114.35056000060.3"/>
<item key="(9999,9995)" value=""/>
<item key="(9999,9997)" value="5"/>
<item key="(9999,9999)" value="\ser1055846208920"/>
</row>
</osc>

```

Fixed-Length-Transformer - OscFixXml

Einleitung

Der Transformer OscFixXml.dll ist eine ActiveX-Dll, mit deren Hilfe Fixed-Length-Daten (Datensätze mit einheitlicher Feldlänge) in ein einheitliches Xml-Format (vgl. Ausgangsformat von OscFixXml) transformiert werden können. Durch dieses einheitliche Format, kann dann sehr bequem aus einem Stylesheet oder einem anderen Programm auf diese Daten zugegriffen werden. Pro Zeile aus dem Quelldokument wird ein Datensatz generiert. Datensatztrennzeichen (z.B. Zeilenumbrüche) werden nicht berücksichtigt. Diese müssen also ein Feld innerhalb des Datensatzes angegeben werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente
Axvbxml.dll	enaio®	enaio® communicator-Komponente
Mxml4.dll	Microsoft	

Installation

Die Komponente OscFixXml.dll wird standardmäßig im bin-Verzeichnis der enaio® communicator-Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein. Diese Komponente wird im bin-Verzeichniss des enaio® communicator abgelegt.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In der Konfiguration wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen

werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rl_Fix_Xml"
      transformer="C"
      definition="OscComTransformer"
      initfile="modules.xml">
  <input messageType="mtFix"/>
  <output messageType="mtFixXml"/>
</rule>
<transformation name="tr_Fix_Xml" rule="rl_Fix_Xml">
  <input supplier="cn_Fix_In" messageType="mtFix"/>
  <output messageType="mtFixXml"/>
</transformation>
```

Außerdem setzt OscFixXml einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="rl_Fix_Xml">
  <!-- Länge der einzelnen Felder, werden jeweils durch Kommas getrennt, Summe
der einzelnen Felder muss die Länge eines Datensatzes sein -->
  <fields>10,10,10,10,10</fields>
  <!-- encoding Deklaration, ISO_8859-1, UTF-8, UTF-16 -->
  <encoding>ISO-8859-1</encoding>
  <!-- Angabe der Felder muss der Anzahl der Felder in fields entsprechen-->
  <fieldnames>eins,zwei,drei,vier,fünf</fieldnames>
  <!-- ProglD der Komponente -->
  <ComProgId>OscFixXml.Trans</ComProgId>
</module>
```

Als Mindestkonfiguration werden folgende Angaben benötigt:

fields

ComProgId

Ist dem Attribut fields wird ein einzelner Datensatz in der Form beschrieben, dass für alle Felder die jeweilige Länge mit Komma getrennt angegeben wird. Die Summe dieser Längen ergibt die Gesamtlänge eines Datensatzes.

In dem Attribut fieldnames können die Bezeichner der einzelnen Felder mit Komma getrennt angegeben werden. Die Anzahl der Namen muss mit der Anzahl der konfigurierten Felder in fields korrespondieren.

Mit encoding kann die Xml-Deklaration für den Output gesteuert werden. Standardmäßig wird hier ab „ISO-8859-1“ gesetzt.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Die weitere Verarbeitung wird meistens mit XSLT-Transformationen innerhalb des enaio® communicator realisiert.

Ausgangsformat von OscFixXml

Aus dem folgenden Schema kann der Aufbau des generierten Xml-Formates des Transformers entnommen werden. Das Format von OscCsvXml entspricht dem der Komponente OscFixXml.Dll. Einziger Unterschied ist, dass bei OscCsvXml im Root-Tag zusätzlich das Attribut sourcefile angegeben werden kann.

OscFixXml ermittelt den Dateinamen, wenn die Option header = filename von OscConFile konfiguriert wurde. Der Dateiname wird dann im Attribut file-name des ROOT-Tag osc abgelegt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
```



```

<xs:element name="osc">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="row" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="data-format" type="xs:string" use="optional"/>
    <xs:attribute name="processing" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="row">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="position" type="xs:int" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="item">
  <xs:complexType>
    <xs:attribute name="key" type="xs:string" use="required"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Die schematische Darstellung (Bild 15):

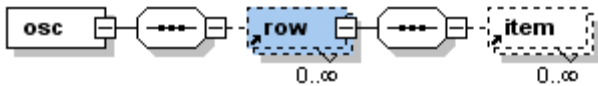


Abbildung 22

Bild 1

Aufbau Fix-Xml

Ein Beispieldatensatz für Fix-Xml:

```

<?xml version="1.0"?>
<osc data-format="fix-length" processing="ISO-8859-1">
  <row>
    <item key="field0" value="Name"/>
    <item key="field1" value="Alter"/>
    <item key="field2" value="Geschlecht"/>
    <item key="field3" value="Wohnort"/>
    <item key="field4" value="Nummer"/>
  </row>
</osc>

```

Beispiele für Konfigurationen und ihre Transformationsergebnisse

Mit Feldnamen

Konfiguration:

```
<module name="fixtoxml">
  <encoding>ISO-8859-1</encoding>
  <fields>10,10,10,10,10</fields>
  <fieldnames>Name,Alter,Geschlecht,Wohnort,Nummer</fieldnames>
  <ComProgId>OscFixXml.Trans</ComProgId>
</module>
```

Dateninput:

Müller....75.....männlich..Berlin....10000.....Müller....75.....männlich..Berlin....10000.....

Ergebnis:

```
<?xml version="1.0"?>
<osc data-format="fix-length" processing="ISO-8859-1">
  <row>
    <item key="Name" value="Müller"/>
    <item key="Alter" value="75"/>
    <item key="Geschlecht" value="männlich"/>
    <item key="Wohnort" value="Berlin"/>
    <item key="Nummer" value="10000"/>
  </row>
  <row>
    <item key="Name" value="Müller"/>
    <item key="Alter" value="75"/>
    <item key="Geschlecht" value="männlich"/>
    <item key="Wohnort" value="Berlin"/>
    <item key="Nummer" value="10000"/>
  </row>
</osc>
```

Ohne Feldnamen

```
<module name="fixtoxml">
  <encoding>ISO-8859-1</encoding>
  <fields>10,10,10,10,10</fields>
  <ComProgId>OscFixXml.Trans</ComProgId>
</module>
```

Dateninput:

Müller....75.....männlich..Berlin....10000.....Müller....75.....männlich..Berlin....10000.....

Ergebnis:

```
<?xml version="1.0"?>
<osc data-format="fix-length" processing="ISO-8859-1">
  <row>
    <item key="field0" value="Müller"/>
    <item key="field1" value="75"/>
```

```

<item key="field2" value="männlich"/>
<item key="field3" value="Berlin"/>
<item key="field4" value="10000"/>
</row>
<row>
<item key="field0" value="Müller"/>
<item key="field1" value="75"/>
<item key="field2" value="männlich"/>
<item key="field3" value="Berlin"/>
<item key="field4" value="10000"/>
</row>
</osc>

```

Der Unterschied zu dem Ergebnis „Mit Feldnamen“ besteht lediglich darin, dass in den key-Attributen die Bezeichner aus der Konfiguration übernommen werden.

LDT/BDT-Transformer - OscLDTML

Einleitung

Der Transformer OscLDTML.Dll ist eine ActiveX-Dll, mit deren Hilfe BDT/LDT-Daten in ein einheitliches Xml-Format transformiert werden können. BDT/LDT ist ein Protokoll für den Austausch von Daten im medizinischen Bereich. Durch die Transformation in ein einheitliches XML-Format, kann dann sehr bequem aus einem Stylesheet oder einem anderen Programm auf diese Daten zugegriffen werden. Mit dem XML-Format wird der Aufbau des Datenpakets nachgebildet und über die bekannten Bezeichner (z.B. FK3001) aus der LDT-Spezifikation der Kassenärztlichen Bundesvereinigung vom 15.08.2003 kann auf die Daten zugegriffen werden. Da es keinen allgemein anerkannten Standard für die Darstellung von LDT-daten in XML gibt wurde hier ein

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente
Axvbxml.dll	enaio®	enaio® communicator-Komponente
Msxml4.dll	Microsoft	

Installation

OscLDTML.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden. Die Komponente axvbxml.dll muss ebenfalls auf dem System registriert sein.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```

<rule name="rl_ldt_Xml"
transformer="C"
definition="OscComTransformer"

```

```

    initfile="modules.xml">
    <input messageType="mtLdt"/>
    <output messageType="mtLdtXml"/>
</rule>
<transformation name="tr_Idt_Xml" rule="rl_Idt_Xml">
    <input supplier="cn_Ldt_In" messageType="mtLdt"/>
    <output messageType="mtLdtXml"/>
</transformation>

```

Außerdem OscLDTML setzt einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```

    <module name="rl_Csv_Xml">
        <!-- Feldnamen stehen in der Datei (erste oder zweite Zeile in Abhängigkeit von
filename, Eingabe 0 oder 1 -->
        <use-paragraphs>true</use-paragraphs>
        <!-- ProgId der Komponente -->
        <ComProgId>OscLDTML.Trans</ComProgId>
    </module>

```

Als Mindestkonfiguration werden folgende Angaben benötigt:

ComProgId

Mit dem Attribut `use-paragraphs` kann gesteuert werden, dass Absätze aus übergebenem Fließtext erhalten bleiben.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Die weitere Verarbeitung wird meistens mit XSLT-Transformationen innerhalb des enaio® communicator realisiert.

Ein- und Ausgangsformat

An dem folgendem Beispiel-XML wird das Ausgangsformat dieses Transformiers gezeigt.

Die Einstellung `use-paragraphs` wurde auf `true` gesetzt. Dadurch wurden alle Zeilen des Bereichs FK8480 jeweils in einem eigenen CRLF-Tag abgelegt.

```

<?xml version="1.0"?>
<XDTML>
  <DATENPAKET>
    <LDPH8220>
      <FK8100>239</FK8100>
      <!--Satzlänge Länge=5, Typ=n-->
      <FK9211>07/99</FK9211>
      <!--Version der Satzbeschreibung Länge=11, Typ=a-->
      <FK0201>000000</FK0201>
      <!--Arztnummer Länge=7 bzw. 9, Typ=n-->
      <FK0203>Prof.Dr.Hempel</FK0203>
      <!--Arztname (Praxisbezeichnung) Länge=60, Typ=a-->
      <FK0204>kA</FK0204>
      <!--Kein Eintrag gefunden-->
      <FK0205>Arnold-Heller-Str. 7</FK0205>
      <!--Straße der Praxisadresse Länge=60, Typ=a-->
      <FK0206>D-12456 Kiel</FK0206>
      <!--PLZ und Ort der Praxisadresse Länge=60, Typ=a-->
      <FK8300>Pathologisches Institut Kiel</FK8300>
      <!--Labor Länge=60, Typ=a-->
      <FK0101>T000000</FK0101>
      <!--KBV-Prüfnummer Länge=8, Typ=a, L=Import LG-Befunddatei M=Import Laborfacharzt-
Befunddatei/Sonstige Einsendepraxis-Befunddatei N=Export LG-Auftragsdatei O=Export

```

Elektronische Überweisung T=Export Sonstige Einsendepraxis Y=Export Laborfacharzt V=Import
Elektronische Überweisung X=Export LG-Befunddatei X=Import LG-Auftragsdatei-->

<FK9106>1</FK9106>

<!--Verwendeter Zeichensatz Länge=1, Typ=n, 1=7-Bit-Code 2=IBM-Code 3=ISO 8859-1

Code-->

<FK8312>000</FK8312>

<!--Kunden- (Arzt-)Nummer Länge=8, Typ=a-->

<FK9103>09080001</FK9103>

<!--Erstellungsdatum Länge=8, Typ=d-->

</LDPH8220>

<SE8204>

<!--Satzart Länge=4, Typ=n, 8201=Labor-Facharzt-Bericht 8202=LG-Bericht

8203=Mikrobiologie-Bericht 8204=Facharzt-Bericht Sonstige Einsendepraxen 8218=Elektronische

Überweisung 8219=auftrag an eine Laborgemeinschaft 8220=L-Datenpaket-Header 8221=L-

Datenpaket-Abschluss 8230=P-Datenpaket-Header 8231=P-Datenpaket-Abschluss 8240=ELV-

Headersatz 8241=ELV-EndeSatz 8242=ELV_Stammsatz 8243=ELV-Löschsatz-->

<FK8100>1440</FK8100>

<!--Satzlänge Länge=5, Typ=n-->

<FK8310>1000882119</FK8310>

<!--Anforderungs-Ident Länge=13, Typ=a-->

<FK8311>E/01/013564</FK8311>

<!--Auftragsnummer des Labors Länge=30, Typ=a-->

<FK8301>02072001</FK8301>

<!--Eingangsdatum des Auftrags im Labor Länge=8, Typ=d-->

<FK8302>05072001</FK8302>

<!--Berichtsdatum Länge=8, Typ=d-->

<FK3101>schmidt</FK3101>

<!--Name des Patienten Länge=28, Typ=a-->

<FK3102>Holger</FK3102>

<!--Vorname des Patienten Länge=28, Typ=a-->

<FK3103>12121940</FK3103>

<!--Geburtsdatum des Patienten Länge=8, Typ=d-->

<FK8401>E</FK8401>

<!--Befundart Länge=1, Typ=a, E=Endbefund T=Teilbefund V=Vorläufige Befund A=Archiv-
Befund N=Nachforderung-->

<FK8407>1</FK8407>

<!--Geschlecht des Patienten Länge=1, Typ=n, 0=unbekannt 1=männlich 2=weiblich 3=Kind
4=Kind, männlich 5=Kind, weiblich 6=Tier-->

<FK8430>Schnellschnitt, Peritoneal-PE</FK8430>

<!--Probenmaterial-Bezeichnung Länge=60, Typ=a-->

<FK8480>

<CRLF>

Makroskopie: 10x bis 7 cm großes dreieckförmiges </CRLF>

<CRLF>

ter liegende, bis 1 cm breitem Fettgewebe. Auf de</CRLF>

<CRLF>

che Herde teils mit zentraler Einziehung, maximal</CRLF>

<CRLF/>

<CRLF>

Schnellschnittdiagnose: (02.07.01). Peritoneum mi</CRLF>

<CRLF>

rose ohne Entzündung und Mesothelhyperplasie. Sic</CRLF>

<CRLF>

nachzuweisen (Befunddurchgabe von Frau Dr. Lüttge</CRLF>

<CRLF/>

<CRLF>

Mikroskopie: (3 Blöcke, je HE, 1 x PAS, 1 x Schne</CRLF>

<CRLF>

Herdförmige peritoneale Bindegewebsvermehrung mit</CRLF>

<CRLF>

perplasie. In den bindegewebigen Plaques ganz ver</CRLF>

<CRLF>

niert verteilte kleine Tumorzellen mit exzentrisch</CRLF>

<CRLF>

sitivem Zytoplasma, teils mit charakteristischem </CRLF>

<CRLF/>

<CRLF>

Diagnose: Peritonealexzistat mit Infiltration durc</CRLF>

<CRLF>

hochgradiger lokaler Stromadesmoplasie.</CRLF>

<CRLF/>

<CRLF>

Kommentar: Wir werden zur Sicherung der Diagnose </CRLF>

<CRLF>

Untersuchung durchführen. Hierüber erfolgt ein ge</CRLF>

<CRLF/>

<CRLF/>

```

<CRLF/>
<CRLF> Prof. Dr. Hempel Dr. Franz </CRLF>
</FK8480>
<!--Ergebnis-Text Länge=60, Typ=a-->
</SE8204>
</DATENPAKET>
</XDTML>

```

Dieses Xml-Dokument wurde aus folgendem LDT-Datensatz generiert.

01380008220

014810000239

014921107/99

005

01602010000000

0230203Prof.Dr.Kremer

0110204kA

0290205Arnold-Heller-Str. 7

0210206D-24105 Kiel

0378300Pathologisches Institut Kiel

0170101T0000000

01091061

0128312000

017910309080001

01380008204

014810001440

01983101000882119

0208311E/01/013564

017830102072001

017830205072001

0163101F|rtsch

0153102Holger

017310312121940

0108401E

01084071

0388430Schnellschnitt, Peritoneal-PE

0698480 Makroskopie: 10x bis 7 cm gro~es dreieckf|rmiges

0698480 ter liegende, bis 1 cm breitem Fettgewebe. Auf de

0698480 che Herde teils mit zentraler Einziehung, maximal

0208480

0698480 Schnellschnittdiagnose: (02.07.01). Peritoneum mi

0698480 rose ohne Entz}ndung und Mesothelhyperplasie. Sic

0698480 nachzuweisen (Befunddurchgabe von Frau Dr. L}ttge

0208480

0698480 Mikroskopie: (3 Bl|cke, je HE, 1 x PAS, 1 x Schne

0698480 Herdf|rmige peritoneale Bindegewebsvermehrung mit

0698480 perplasie. In den bindegewebigen Plaques ganz ver

0698480 niert verteilte kleine Tumorzellen mit exzentrisch

0698480 sitivem Zytoplasma, teils mit charakteristischem

0208480

0698480 Diagnose: Peritonealexzistat mit Infiltration durc

0598480 hochgradiger lokaler Stromadesmoplasie.

0208480
 0698480 Kommentar: Wir werden zur Sicherung der Diagnose
 0698480 Untersuchung durchf}hren. Hier}ber erfolgt ein ge
 0208480
 0208480
 0208480
 0698480 Prof. Dr. Kl}ppel Dr. Franz

OscTrFallbackrouter

Einleitung

Der Transformer OscTrFallbackRouter.Dll ist eine ActiveX-Dll. Dieser Transformer ermöglicht es beliebig viele Ausgangskanäle der Reihenfolge nach abzuarbeiten, bis die zu verarbeitende Nachricht verarbeitet werden konnte. Zum Beispiel könnte so versucht werden eine Nachricht, die von dem Import abgelehnt wird an eine Komponente zu versenden, die die Nachricht aufbereitet und anschließend den Import neu zu versuchen.

Beispiel:

1. OscTrFallbackRouter bekommt eine Importnachricht und versendet diese an OscConImpE (vgl. Dokumentation)
2. OscConImpE kann die Nachricht nicht importieren, weil der Patient nicht gefunden werden kann.
3. OscTrFallbackRouter sendet die Nachricht an einen Transformer, der diese Nachricht aufbereitet und mit einer Standard-ID versieht, die einem Dokument gehört auf das alle Dokumete importiert werden, die nicht zugeordnet werden können.
4. OscComImpE kann die Nachricht nun importieren.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

OscTrFallbackRouter.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rl_jivex_router"
      transformer="C"
      definition="OscComTransformer"
      initfile="modules.xml">
  <input messagetype="mt_osimport"/>
  <output messagetype="mt_osimport"/>
```

```

    <output message="mt_job_fallback"/>
  </rule>
  <transformation name="tr_jivex_router" rule="rl_jivex_router">
    <input supplier="trImpJiveX" message="mt_osimport"/>
    <output message="mt_osimport"/>
    <output message="mt_job_fallback"/>
  </transformation>

```

Außerdem setzt OscTrFallbackRouter einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```

<module name="rl_jivex_router">
  <!-- In der abzuarbeitenden Reihenfolge die topic angeben, mit Komma separiert-->
  <topics>mt_osimport@tr_jivex_router,mt_job_fallback@tr_jivex_router</topics>
  <ComProgId>OscTrFallbackRouter.Trans</ComProgId>
</module>

```

Als Mindestkonfiguration werden folgende Angaben benötigt:

ComProgId

topics

In dem Attribut topics muss angegeben werden, in welcher Reihenfolge mögliche Nachrichtentypen versendet werden sollen. Abonnenten dieser Nachrichten werden dann automatisch mit der versendeten Nachricht versorgt und können diese dann verarbeiten. Solange die Nachricht nicht erfolgreich verarbeitet werden konnte, versendet OscTrFallbackRouter diese Nachrichten, bis kein Abonnent mehr vorhanden ist.

Da die Konfigurationsdateien Xml-Dateien sind, wird auf die korrekte Groß-/Kleinschreibung geachtet.

Die weitere Verarbeitung wird meistens mit XSLT-Transformationen innerhalb des enaio® communicator realisiert.

OscTrLookUp

Einleitung

Der Transformer OscTrLookUp.Dll ist eine ActiveX-Dll. Dieser Transformer verarbeitet Anfragen an die Archivdatenbank die nach dem Schema von DMSQuery (vgl. dazu Aufbau von Anfragedatensätzen) entsprechen. Dadurch können Exporte aus der Archivdatenbank realisiert werden und Daten aus der Archivdatenbank über weitere Komponenten des enaio® communicators verarbeitet werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Mxml4.dll	Microsoft	
Oxmljsc.Dll	enaio®	Server-Komponente
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

OscTrLookUp.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation

den Archiv-Server angegeben. Das Passwort muss kodiert angegeben werden. Dafür wird das Tool OscPWDCrypter verwendet.

Das Attribut `file-props` wird in der derzeitigen Version der Komponente noch nicht verwendet und ist für den späteren Gebrauch reserviert.

Mit `tmp-directory` wird das temporäre Verzeichnis angegeben, das durch die Komponente für das Zwischenspeichern von Dateien verwendet wird.

In `encoding` wird angegeben, welche Kodierung das Anfrageergebnis erhalten soll. Derzeit wird nur utf-8 korrekt verarbeitet.

In `replace-query` wird angegeben, dass die Anfrageknoten aus dem Dokument entfernt werden sollen.

In `convert-tif-to-pdf` wird angegeben, dass TIFF-Bilder in PDF-Dokumente gewandelt werden.

In `split-dmscontent-to-topic` wird angegeben, dass die Ergebnisse einzeln an das hier konfigurierte Topic weitergeleitet werden.

In `set-fileextension-from-mimetype` wird angegeben, dass die Dateierweiterung explizit nach dem gelieferten Mimetype gesetzt wird. Derzeit wird nur JPEG unterstützt.

Aufbau von Anfragedatensätze

Es werden Anfragen entsprechend dem Schema DMSQuery verarbeitet. Dieses muss zu der entsprechenden Version zu der Komponente oxjobdms konform sein. Es werden alle HOL und LOL-Anfragen akzeptiert.

Es ist möglich einzelne Anfrage oder mehrfache Anfragen mit einem Anfragedatensatz zu stellen.

Ein Beispiel für eine einfache Anfrage ist nachfolgend zu sehen.

```
<?xml version="1.0"?>
<DMSQuery requesttype="HOL" fileinfo="1" status="1" baseparams="1">
  <Archive name="Patient_MIC">
    <ObjectType name="Patient_MIC">
      <Fields field_schema="ALL"/>
      <Conditions>
        <ConditionObject name="Patient_MIC">
          <FieldCondition name="Patienten-Nr.">
            <Value>03210773</Value>
          </FieldCondition>
        </ConditionObject>
      </Conditions>
      <ChildObjects child_schema="ALL" export_depth="10"/>
    </ObjectType>
  </Archive>
</DMSQuery>
```

Es können auch mehrere Anfragen innerhalb eines Anfragedatensatzes gestellt werden. Nachfolgend ein Beispiel.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<osc>
  <kn>
    <ir>
      <DMSQuery requesttype="HOL">
        <Archive name="Patient_MIC">
          <ObjectType name="Patient_MIC">
            <Fields field_schema="ALL"/>
            <Conditions>
              <ConditionObject name="Patient_MIC">
```

```

        <FieldCondition name="Patienten-
Nr.">
            <Value>1245679</Value>
        </FieldCondition>
    </ConditionObject>
</Conditions>
<ChildObjects child_schema="ALL"
export_depth="10"/>
    </ObjectType>
</Archive>
</DMSQuery>
</ir>
</kn>
<meinedaten>
    <DMSQuery requesttype="HOL">
        <Archive name="Patient">
            <ObjectType name="Patient">
                <Fields field_schema="ALL"/>
                <Conditions>
                    <ConditionObject name="Patient_MIC">
                        <FieldCondition name="Patienten-Nr.">
                            <Value>1245690</Value>
                        </FieldCondition>
                    </ConditionObject>
                </Conditions>
                <ChildObjects child_schema="ALL" export_depth="10"/>
            </ObjectType>
        </Archive>
    </DMSQuery>
</meinedaten>
</osc>

```

Die Komponente OscTrLookUp interpretiert alle DMSQuery-Tags als Anfragefragment und versucht damit Anfragen an den Archivserver zu stellen.

Aufbau von Ergebnisdatensätze

Die Ergebnisdatensätze entsprechen dem Aufbau der Anfragedatensätze. Es werden jedoch in der derzeitigen Implementation der Komponente die Anfragen (DMSQuery-Tags) nach der Bearbeitung durch XML-Daten, die dem Schema DMSContent entsprechen, ersetzt. Beispielhaft ist folgendes Ergebnis.

```

<DMSContent format="HOL" version="4.50.571.4112" timestamp="2004-04-23T09:17:37"
user="SCHUSTER" station="JSCHUSTER">
    <Archive name="Patient_MIC" id="9" osguid="84AFC358F42A4FA69CACA8D72491C931">
        <ObjectType name="Patient_MIC" id="9" osguid="84AFC358F42A4FA69CACA8D72491C931">
            internal_name="Patient" type="FOLDER" table="stamm10">
                <ObjectList>
                    <Object id="110792">
                        <Fields>
                            <Field name="Patienten-Nr." datatype="TEXT"
dbname="feld1" ostype="X" size="10">1245679</Field>
                            <Field name="Name" datatype="TEXT"

```

```

dbname="feld2" ostype="X" size="94">Mic</Field>
      <Field name="Vorname" datatype="TEXT"
dbname="feld3" ostype="X" size="94">PATient</Field>
      <Field name="Geburtsname" datatype="TEXT"
dbname="feld4" ostype="X" size="30"/>
      <Field value="M" name="Geschlecht"
datatype="TEXT" dbname="feld5" ostype="G" size="1"> männlich</Field>
      <Field name="Geburtsdatum" datatype="DATE"
dbname="datum1" ostype="D" size="10">16.04.2004</Field>
      <Field name="Stra  e" datatype="TEXT"
dbname="feld6" ostype="X" size="106">Berliner Stra  e 100</Field>
      <Field name="PLZ" datatype="TEXT" dbname="feld7"
ostype="X" size="5">12623</Field>
      <Field name="Wohnort" datatype="TEXT"
dbname="feld8" ostype="X" size="106">Berlin</Field>
      <Field value="M" name="Familienstand"
datatype="TEXT" dbname="feld9" ostype="X" size="12"> verheiratet</Field>
      <Field name="Telefon" datatype="TEXT"
dbname="feld10" ostype="Z" size="30">0124645</Field>
      <Field name="Angeh  rige" datatype="TEXT"
dbname="feld11" ostype="X" size="50">Keine</Field>
    </Fields>
  </Object>
</ObjectList>
<Statistics startpos="0" pagesize="-1" total_hits="1"/>
</ObjectType>
</Archive>
<Messages/>
</DMSContent>

```

Werden Dateien zu Dokumenten gefunden, werden diese unterhalb des FileProperties-Tags des DMSContent-Datensatz als Dateiverweis und zus  tzlich Base64 kodiert eingebettet abgelegt.

```

<FileProperties count="1" size="5067" extension=".png" mimetype="image/png">
  <Files>
    <File path="D:\osc-
dist\data\MIC\tmp\jp4_904E9311E35949D9A4E0F73927D51F0C.000">iVBORw0KGgoAAAANSUUhEU
gAAADAAAAAwCAYAAABXAvmHAAAk3RFWHRDcmVhdGlvbiBUaW1AERpIDE2IERleiAyMDAzID
E3OjA0OjI2ICswMTAwYDYt7gAAAAd0SU1FB9MKBhQXl4tufRQAAAAJcEhZcwAACvAAAARwAUKsN
JgAAAEZ0FNQQAAsY8L/GEFAAANOKIEQVR42r1aeWwU5xv/c+x6vT7WB15jMNjGxAmpQwDRpF
WiFhCUuqVKldCqjSLlYBRtVH7R9WqJQKkpEWtEilS21RUFVHTJiFCSXDUHFyJuGxwbYMBL8b3bS9
ee9e73vXszM7V92Z2vlfXVwiM9DQ7O7Pf93vv/d41NgO3eRw/7sm220tqRVG4X1W1KI2HNQCqS9dZ
u6bJjKYpkqrK05IUG5GkmcFQKNjV3Hy...
    </File>
    <File path="D:\osc-
dist\data\MIC\tmp\jp4_FAB097E79906454B86507BBDBB972F2D.DIA">SUkqAEIEAACAP+BQOCQ
WDQeEQmFQuGQ2HQ+IRGJROKRWLRmMRmNRuOR2PR+QSGRSODO+SRldr2RquFv1VKUvLpE
SaOqpnw1+Hc5BEJlHeq+NJlvxB8ncwAEIAGgRduqeJvoAAKaRZ8iebxF5gAHR...
    </File>
  </Files>
</FileProperties>

```

Durch diese Form der Kodierung ist es m  glich Nachrichten   ber Dom  nengrenzen zu transportieren, die Dokumente mit Dateien enthalten. Diese k  nnen dann wieder hergestellt werden und verarbeitet werden.

OscTrBase64

Einleitung

Der Transformer OscTrBase64.Dll ist eine ActiveX-Dll. Dieser Transformer kodiert/dekodiert aus/in Base64 entsprechend der Konfiguration. Es können gesamte Datenströme bzw. XML-Textknoten kodiert/dekodiert werden. Werden Base64-Daten innerhalb von XML-Daten dekodiert, werden diese Textknoten durch Dateireferenzen ersetzt. Der umgekehrte Weg ist, dass in einem XML-Knoten eine Dateireferenz übergeben wird und dies durch Base64-kodierte Daten ersetzt werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®. Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Msoxml4.dll	Microsoft	
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

OscTrLookUp.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert. Da es sich um eine ActiveX-Dll muss diese mit dem Tool regsvr32.exe registriert werden.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rlBase64"
  transformer="C"
  definition="OscComTransformer"
  initfile="Osc.Modules.xml">
  <input message="mtBase64"/>
  <output message="mtBase64"/>
</rule>
<transformation name="trBase64" rule="rlBase64">
  <input supplier="trSupplier" message="mtBase64"/>
  <output message="mtBase64"/>
</transformation>
```

Außerdem setzt OscBase64 einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="rlBase64">
  <!--Steuerung des Verhaltens des Transformers -->
  <action>xml-decode</action>
  <!--In dieses Verzeichnis werden extrahierte Daten als Datei gespeichert und können
dann später beispielsweise im Import genutzt werden. Im Dokument wird ein Dateiverweis angelegt -->
  <directory>c:\temp</directory>
  <!--XPath-Ausdruck, Knoten dieser Liste wird kodiert bzw. dekodiert -->
  <xpath>//base64</xpath>
  <!--Ist dieser Parameter gesetzt, erhält die decodierte Datei die Extension PDF-->
```

```
<fileext>PDF</fileext>
```

dem XPath -->

```
<!--Ist dieser Parameter gesetzt, dann erhält die decodierte Datei die Extension aus
```

```
<fileext-xpath>/*/*/*[OBX/OBX.5/CE.4='Base64']/OBX/OBX.4</fileext-xpath>
```

```
<ComProgId>OscTrBase64.Trans</ComProgId>
```

```
</module>
```

Beschreibung der Parameter

action

Festlegung, wie die Eingangsdaten transformiert werden sollen.

Mögliche Werte:

```
<action>xml-encode</action>
```

Kodieren der xml-Daten nach Base64

```
<action>string-encode</action>
```

Kodieren der String-Daten nach Base64

```
<action>xml-decode</action>
```

Dekodieren der xml-Daten von Base64

```
<action>string-decode</action>
```

Dekodieren der String-Daten von Base64 in eine Datei. Der kodierte Inhalt des base64-Knotens wird durch den Dateinamen ersetzt.

xpath

XPath-Ausdruck, der die Knoten beschreibt, die die zu transformierenden Daten enthalten.

Beispiel:

```
<xpath>//base64</xpath>
```

directory

In diesem Verzeichnis werden alle dekodierten Daten als Datei gespeichert. Der Dateiname besteht aus einem Timestamp. Die Datei-Extension wird durch **fileext** bzw. **fileext-xpath** festgelegt.

Beispiel:

```
<directory>c:\temp</directory>
```

fileext

Festlegung der Extension für die dekodierte Datei.

Ist dieser Parameter nicht festgelegt, dann wird als Extension „.dat“ verwendet.

Beispiel:

```
<fileext>PDF</fileext>
```

fileext-xpath

Festlegung der Extension für die dekodierte Datei durch einen XPath –Ausdruck

Beispiel:

```
<fileext-xpath>/**/*/*[OBX/OBX.5/CE.4='Base64']/OBX/OBX.4</fileext-xpath>
```

Die Reihenfolge, mit der die Dateiendung der dekodierten Datei ermittelt wird, ist folgende:

1. Ist **fileext-xpath** gesetzt, wird dieser Wert genommen.
2. Ist **fileext** gesetzt, wird dieser Wert genommen.
3. Sonst wird die Endung 'dat' verwendet.

Als Mindestkonfiguration werden folgende Angaben benötigt:

Für XML-Daten : **action** und **xpath**. Bei Dekodierung auch **directory**

Für String-Daten: **action**

Folgende Kombinationen der Parameter sind möglich:

1.

```
<action>xml-encode</action>
```

```
<xpath>//base64</xpath>
```

Es werden alle Textknoten, die auf den XPath-Ausdruck matchen nach Base64 kodiert.

2.

```
<action>xml-decode</action>
```

```
<xpath>//base64</xpath>
```

```
<directory>c:\temp</directory>
```

Es werden alle Textknoten, die auf den XPath-Ausdruck matchen Base64 dekodiert und durch einen Dateiverweis ersetzt.

3.

```
<action>xml-encode</action>
```

Der übergebene Datenstrom wird Base64 kodiert.

4.

```
<action>xml-decode</action>
```

Der übergebene Datenstrom wird Base64 dekodiert.

5.

```
<action>string-encode</action>
```

Die übergebene Zeichenkette wird Base64 kodiert.

6.

```
<action>string-decode</action>
```

Die übergebene Zeichenkette wird Base64 dekodiert.

OxTrMuse

Einleitung

Dieser Transformierer extrahiert PDF-Daten aus einer übergebenen HL7-Nachricht aus dem Muse-System.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Installation

OxTrMuse.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```
<rule name="rIMuse"
  transformer="C"
  definition=" oxtrmuse"
  initfile="modules.xml">
  <input messagetype="mtMuse"/>
  <output messagetype="mtMuse"/>
</rule>
<transformation name="trMuse" rule="rIMuse">
  <input supplier="trSupplier" messagetype="mtMuse"/>
  <output messagetype="mtMuse"/>
</transformation>
```

Außerdem setzt OxTrMuse einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="rIMuse">
  <!--Verzeichnis, in das die PDFs kopiert werden sollen -->
  <directory>D:\dvp\las450\oscsrver\CPP\data\oxtrmuse\oxtrmuse</directory>
</module>
```

OscTrLabMsgLookUp

Einleitung

Der Transformer OscTrLabMsgLookUp.Dll ist eine ActiveX-Dll. Dieser Transformer kann innerhalb


```

<!-- wenn bereits eine vorhandene Nachricht mit der Message-ID der -->
<!-- aktuellen NAchricht in der TAbelle Itt_obsresults gefunden wird -->
<messageid-loglevel>WARN</messageid-loglevel>
<ComProgId> OscTrLabMsgLookUp.Trans</ComProgId>
</module>

```

messageid-loglevel

Mit dieser Einstellung kann gesteuert werden, wie im Fall, dass ein Nachrichtenduplikat gesendet wird, geloggt werden soll. Es kann zwischen den Einstellungen INFO und WARN gewählt werden.

OscTrDebugPassthrough

Einleitung

Der Transformer OscTrDebugPassthrough.Dll ist eine ActiveX-Dll. Dieser Transformer kann für das interaktive Debugging eingesetzt werden. Wird dieser Transformer in eine Kommunikationsstrecke konfiguriert, können Nachrichten interaktiv eingesehen und bearbeitet werden. Die Komponente ist nur für die Teststellung oder Testläufe zu verwenden. Während der Bearbeitung einer Nachricht können auf diesem Kanal keine weiteren Nachrichten verarbeitet werden.

Voraussetzungen

Die Komponente ist unabhängig von den verschiedenen Versionen von enaio®.

Folgende Komponenten müssen auf dem System vorhanden sein.

Abhängigkeiten		
Osccomdefs.dll	enaio®	enaio® communicator-Komponente

Installation

OscTrDebugPassthrough.Dll wird standardmäßig im bin-Verzeichnis der enaio® communicator Installation installiert.

Konfiguration

Konnektoren bzw. Transformatoren müssen innerhalb der enaio® communicator-Konfiguration an mindestens 2 Stellen konfiguriert werden. Zum einen in der aktiven Konfiguration. In dieser wird dem enaio® communicator bekannt gegeben, welche Komponenten „wie“ geladen werden sollen. Beispielsweise könnte diese Konfiguration folgend aussehen.

```

<rule name="rIDebug"
  transformer="C"
  definition="OscComTransformer"
  initfile="Osc.Modules.xml">
  <input message="mtAny"/>
  <output message=" mtAny "/>
</rule>
<transformation name="trDebug" rule="rIDebug">
  <input supplier="cnInput" message=" mtAny "/>
  <output message=" mtAny "/>
</transformation>

```

Außerdem setzt OscTrDebugPassthrough einige Konfigurationseinstellungen in einem eigenen Konfigurationsmodul (hier wäre das modules.xml) voraus. Folgend werden alle Konfigurationseinstellungen gezeigt:

```
<module name="riDebug">  
  <ComProgId> OscTrDebugPassthrough.Trans</ComProgId>  
</module>
```