DGS-KORPUS

Documentation of the Feedback-System and its Integration into iLex

Text: Sven Berding

Graphics: Sven Berding, Thomas Hanke

Release History: Sven Berding, 2015-02-10 (2.0.2)

2015-03-31: fixes some typos, minor updates (Thomas Hanke)

Table of contents

1. Background

In order to explore the active and passive treasurey of words of German Sign Language (DGS), a crowd-sourcing project has been initiated to complement and verify corpus data of DGS.
Besides an active word pool that is well-known by the community a huge passive treasury of words does exist which is characterized by non-documented signs, sign language dialects and colloquial language.
These low-frequency signs and linguistic phenomena are not explored sufficiently. Supplementary methods are needed to complement and verify available corpus data in the process of dictionary compilation. One central descision of the project was to use methods of crowd sourcing and community sourcing in order to verify and to complement corpus data and supplementary information on signs and sign uses. Two different strategies were combined to get members of the language community involved. One strategy was to choose a focus group while the second strategy was to be targeted on the evaluation of the data pool by online feedback.
The focus group approach had been successfully applied in various dictionary projects before and being a form of community sourcing it was considered a suitable instrument in the project context. In this case qualified community members committed themselves to the project for a longer period thus providing continuity and high quality of work. In our case, the focus group consisted of 10 signers with high language awareness and some metalinguistic knowledge. They discussed specific questions on the use of signs that came up in the dictionary compilation process and that cannot be answered on the basis of available corpus data. This was put into practice by means of introspection and filmed group discussions, resulting in mostly qualitative data.
In order to involve the language community as a whole (crowd sourcing) an online feedback platform is now available which is in focus of this technical documentation for administrators. The online feedback platform is a web application that enables members of the language community to answer questions on signs, their variants, and senses. Results provide evidence for regional distribution of signs and sign meanings. The answers are analysed quantitatively and provide information that will complement and verify data from the corpus and other sources. As with any crowd sourcing approach addressing a rather small community, the crucial point for the Feedback platform is not only how to attract enough first-time users, but also how to make users check back regularly.
In order to achieve this objective, a gamification approach has been implemented by using computer game elements such as high-scores and expert levels that combines well with the target community's pride of their own language and their support of the project.
The feature that is unique to this system compared to the many online survey tools available is its sign language rootedness: Not only does the system address the user in sign language, but the user can provide answers to open questions in sign as well.
This technical documentation gives a profound insight into the Online-Feedback web application, its installation into the Tomcat servlet container[1], configuration issues as well as the creation of questionnaires and the exploration of results.

2. Disambiguation and a first look at the application

This chapter gives a brief overview on how the GUI of the Feedback web application looks like and the appearance of questions in a questionnaire in general. The concepts of the application will be discussed in the following chapters in detail. In order to get quick access to the contents of this documentation, it seems appropriate to present the "core" to the reader for a start. A questionnaire is a bundle of different questions in the sign language context.

[1] http://tomcat.apache.org

Disambiguation

Before this documentation carries on, it is important to know why the term “packidge” has been selected for internal use in favour of “package”. In fact, what we consider here is a bundle of questions that have been organized in a certain way in order to make computational processing of the answers as easy as possible and to present questions to users appropriately. But since DGS-Feedback is a web-application that is based on the Java[2] programming language it is not possible to call a package simply “package” inside of Feedback. This is because the term “package” is a reserved keyword in Java, so it is forbidden to use the term inside the source code in another context than to declare a 'name space' for a Java class. It has to be put at the top of the Java file and it should be the first Java statement line.

In consequence a new term had to be found in order to describe the bundled request construct. The “packidge” term was chosen because it semantically reminds of the original package term and does not cross any Java language specification.

Each single question of a so-called packidge (package, cf. chapter 7) is presented to the user in the web browser as follows:

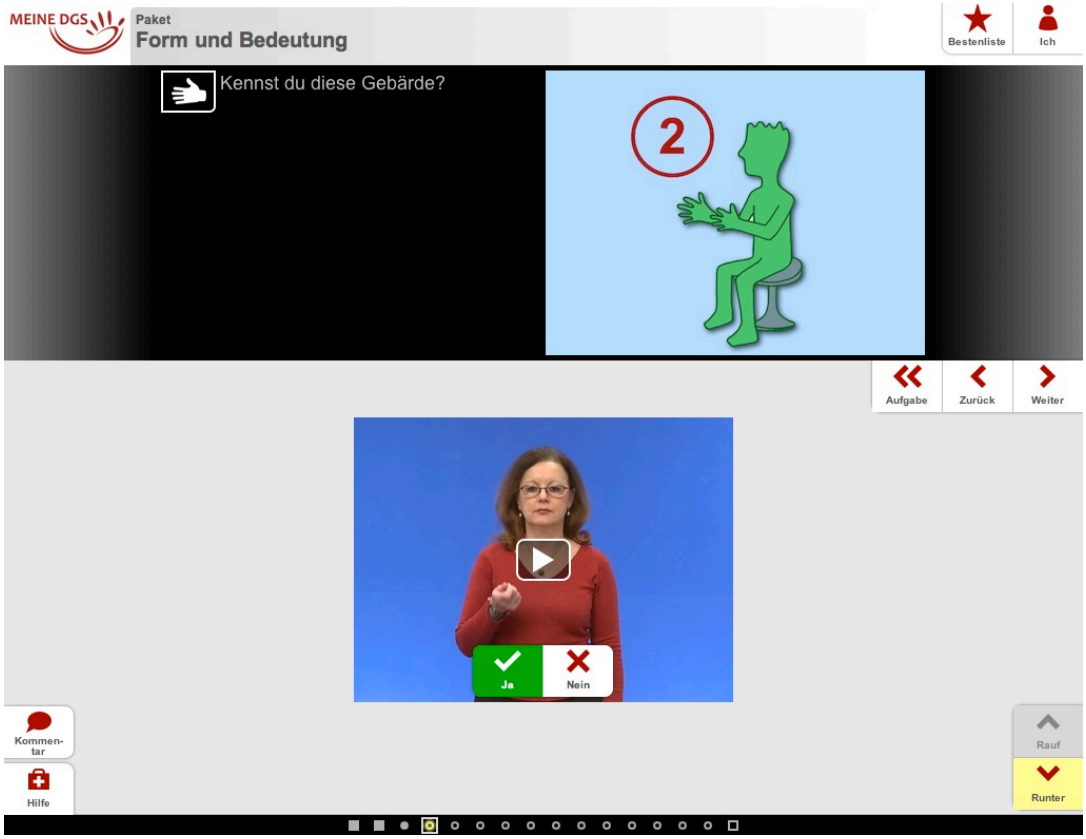


Figure 1: Presentation of a sign

[2] http://www.java.com

Depending on the type of a questionnaire different kinds of questions are presented to the user in the upper window part such as “Do you know this sign?” (Form an meaning bundle). Each question in the upper part of the GUI refers to a video or text contribution that is visible in the bottom window part.
Questions on signs for example refer to videos and texts presented in the following screenshots.

Figure 2: Money - Medium of exchange

Each sign is presented in the left GUI part whereas the textual meaning of the current sign is shown on the right. The user has to select her/his answer by the buttons in the middle part. There will be a lot of examples and discussions on that topic in the progress of the current documentation. The purpose now is just to give you a feeling on how the general GUI look and feel is.
For example, the hand-symbol means: This here is the sign which I usually use. The eye-symbol means: I know this sign but I usually do not use it myself but I’ve seen others using it. The striked-eye symbol means: I don’t know this sign.
As the following screen makes clear it is also possible to present some annotations on the given sign. The annotations are given in the video on the very right position of the window. Since one single sign can have different readings this is an appropriate way to add some details on the context.

Figure 3: Currency - Euro, Dollar

Referring to figure 1 we can see that there is a down button in the GUI which allows the user to navigate from one presented reading of a sign to the next while staying on the same page of the bundle. The subsequent page whereas can be accessed by clicking the “Weiter” button in the upper part of the GUI. The next page will present a different sign and its readings.
The questionnaires generally contain questions on signs and its readings, meanings and regional distribution. A further type refers to handedness. As we will see later on there are also bundles that handle the metadata information of a user (i.e. master file data etc.).
In the present example some questions on a sign are posed. The process is as follows. At first the sign is presented to the user without lip movement. If the user does not know the sign the next sign is presented – again without lip movement whereas if the user knows the sign further questions on the different meanings of the sign are asked. The contextual meanings of the sign are requested with the corresponding lip movement at this time (i.e. "money" and "currency").
Often there are little differences in the hand shape like “knife“ with one finger or “knife“ with two fingers. This does not mean the same in context of the application. On finishing one page the user has the opportunity to add even more meanings by text or video.

Handedness

This type of questionnaire is about which hand is the preferred siging hand. For the appearance of such questionnaire packages, please take a look at the following screen.

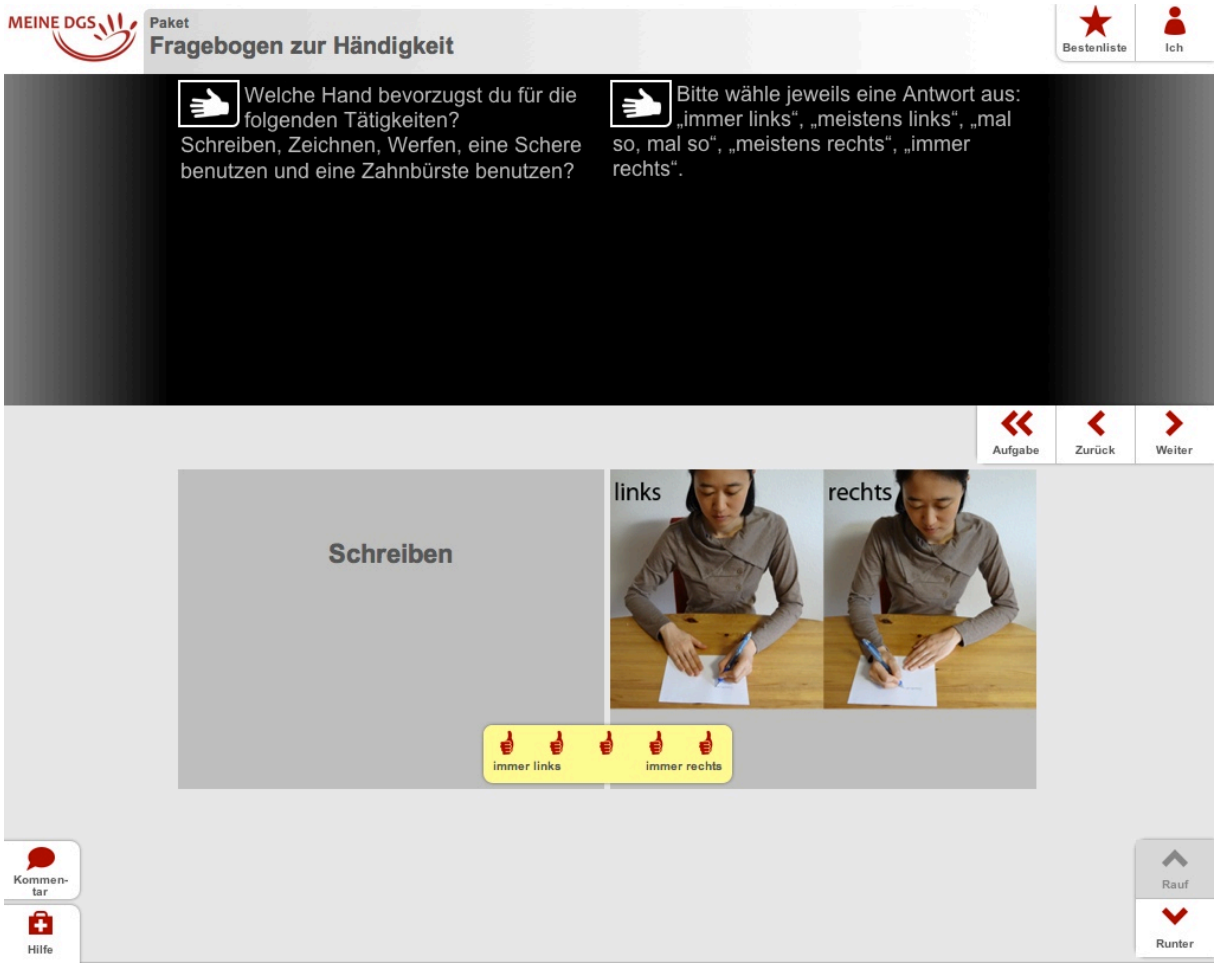


Figure 4: Questionnaire on handedness

This request makes use of a scale for the answering process. Scale questions are covered in chapter 8.4.2.
Regarding the content of a packidge we can identify some other questionnaire types but the handling for the web application and by the user is the same as already described. For example there are further packages on regional differences of a sign. As you will become aware in chapter 8.3 (metadata related questionnaires) master file data is requested by questionnaires as well.
If a user has finished answering a bundle she/he is able to commit the whole packidge by clicking the button on the middle-right in the next screenshot.

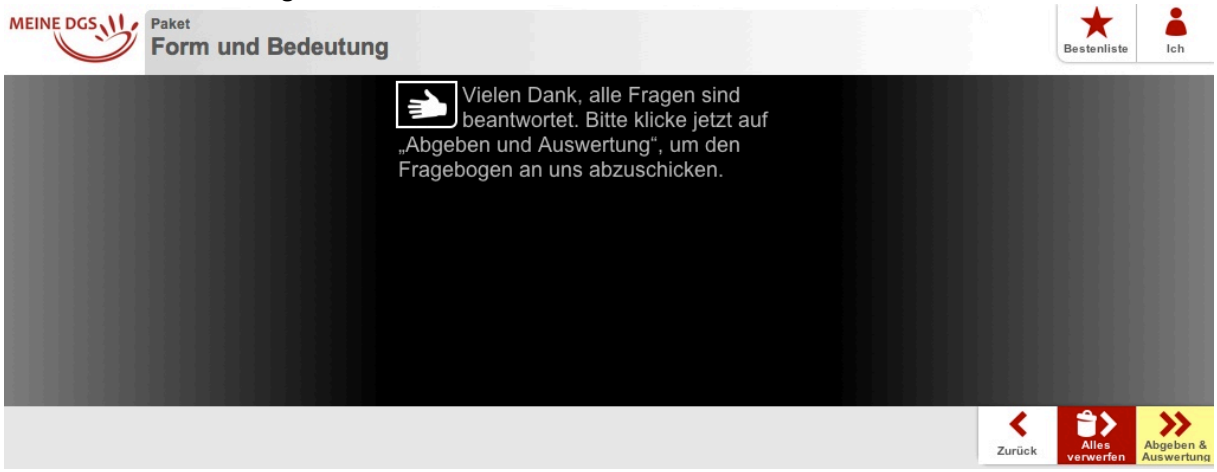


Figure 5: Commit screen

3. About this documentation

Online-Feedback represents a questionnaire-system implemented as a Java web-application. The application makes use of the Google Web Toolkit[3]. GWT itself is a development toolkit for building and optimizing complex browser-based applications. Being an open source set of tools web developers are able to create and maintain complex JavaScript front-end applications in Java.
This is how Online-Feedback provides individual graphical user interfaces (GUI) for PC, Mac and mobile devices. Since the GWT renders the browser-based user interface differently for every mobile target device a responsive design could be implemented.
In addition to a mouse-based navigation (click) Online-Feedback gives the user the opportunity to comment signs by text and video.
The Online Questionnaire System can be integrated with the iLex[4] transcription environment that is to say there is only a loose coupling between the systems on file system level. iLex fetches its data from the file system of the Feedback application and evaluates the XML data with the help of XSLT. Since the data is transmitted in this manner, no data remains in the Feedback file system after the process. This is extremely important for the sake of data security since the Tomcat file system is exposed to the internet as is the nature of a web server.
With regard to the administration of Feedback users, a role mapping strategy has been implemented between the two systems. There are the internal user groups for the Feedback application (roles) while iLex keeps its own user concept differing in functionality. The n:1 mapping between iLex and Feedback groups will be part of chapter 19.1.
This documentation gives information about the following essential questions:

- How to perform the configuration for the Feedback-App in Apache Tomcat
- How to produce content for Feedback / Creation of questionnaires
- Where to find and how to evaluate the results of the answering process
- Integration of Feedback into iLex
- Handling of Feedback artefacts inside of iLex

For a general overview on the iLex transcription environment please take a look at the iLex wiki[5].

4. Installation and configuration of Feedback

Since the application runs in context of sensitive personal data, the aspect of data security is highly prioritized. Therefore it is extremely important that iLex fetches the data from the Tomcat file system in a way that no personal data remains after the process in the file system source that is exposed to the internet. Video files have to be processed over https for security reasons.
Since Tomcat itself does not take care of the video streams, there needs to be a second, separate https server available in the setup that provides the video data which is referenced in the system's xml files. The video data should be available multiple-resolution and multi-format (such as mp4 or webm) in order to make a responsive presentation for different devices possible. As discussed in chapter 4.3.2, an m3u8 playlist file organizes the metadata in the application context.
An administrator has to be aware of the type and quantity of the target systems in order to provide appropriate multimedia files.
iLex fetches the video files referenced in the result packidge in the result packidge (answers to open questions as well as comments) from the Tomcat user's directory, renders it into the mp4 format for further processing and finally deletes the source file from the user's directory as this is person-related data too.

[3] http://www.gwtproject.org/overview.html
[4] http://www.sign-lang.uni-hamburg.de/ilex
[5] https://wiki.sign-lang.uni-hamburg.de/groups/ilex/

Since Feedback is a Java based web-application, the program files are deployed in the Apache Tomcat servlet container. Tomcat acts as a Java specific runtime environment for the application. In order to get the application to work after an install from scratch, some additional technical details have to be considered.
The following parts of this documentation give information about how to configure Tomcat for the Feedback-App generally, which directories have to be readable and writeable, how to address this when creating new users and how to enable Tomcat's SSL/HTTPS features.

4.1 SSL Configuration

In order to make a proper configuration of Tomcat's SSL features you have to be aware of some crucial points such as certificates and keystores. As a prerequisite you have to have a JDK[6] (Java Development Kit) installed. In this context everything is Java based since we are using a Tomcat Web Server.
When Tomcat needs some SSL-Information about its deployed websites it will have a look at its own keystore. This is the place where Tomcat stores all its SSL-Information. The keystore itself is only a file with the extension ".keystore", ".p12" or similar.
When the browser navigates to an https address, it not only looks for a SSL certificate but also checks if the certificate can be trusted. If the certificate does not originate from a CA (Certification Authority – such as VeriSign or GoDaddy etc.) that the browser trusts in, the browser will give a warning message such as the following:

Figure 6: Not trusted connection (Mozilla Firefox)

SSL Certificate Overview

The initial creation of certificates is described by the next figure.

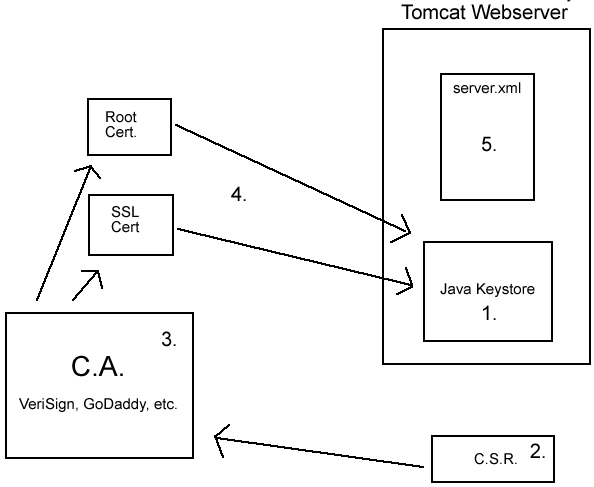


Figure 7: SSL Processing in Tomcat

[6] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Tomcat's SSL-Processing is described according to the figure above. In a second step we will utilize Java's Keytool to produce the initial files.
1. – Creation of keystore
Tomcat's default keystore already contains a self-signed certificate on creation. This is OK for testing setups, but not for production purposes.
2. – Creation of a Certificate Signing Request (C.S.R.)
The CSR is a low encryption key that the CA will require in order to generate the real SSL-Certificates intended for production use. The CSR itself can be created on any machine with a JDK. The CSR is a text file.
Copy and paste the CSR key where the CA wants you to for purchasing your certificate. In academic contexts, computing centres often act as CAs as well.
3. – CA (Certificate Authority) generates the certificate and delivers the files
CAs are reputable companies you can trust. Examples: VeriSign[7], GoDaddy[8] etc. Academic CAs often act as sub-CAs of such companies.
4. – The certificate comes in a bundle of 1 – 3 files containing ROOT CERT an the real SSL CERT

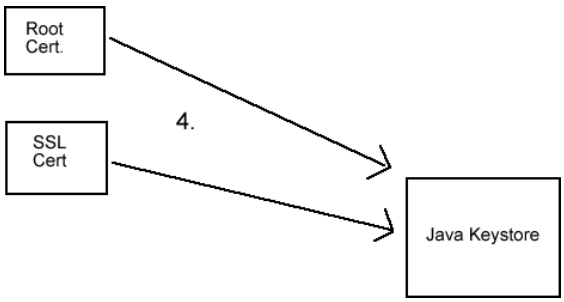


Figure 8: Step 4 – Import certificate into the Tomcat keystore

5. - Modify / Configure server.xml
Tomcat's server.xml is its central configuration file. Located in the /conf folder it contains connector declarations that can be adapted in order to use the keystore.
In order to generate the files in the above-mentioned process you have to follow the following step by step guide.

1. – Create Tomcat keystore in %CATALINA_HOME%\conf with this single command
%JAVA_HOME%\bin\keytool -genkey –alias tomcat –keyalg RSA -keystore %CATALINA_HOME%\conf\my.keystore

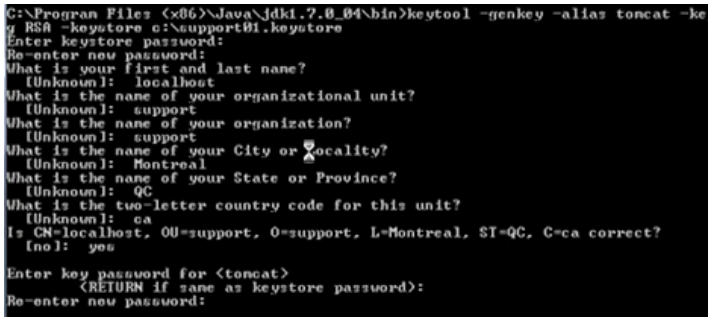


Figure 9: keytool query

[7] http://www.verisign.com
[8] https://de.godaddy.com

By using keytool from the command line you will be prompted to insert some additional information for the keystore as shown in the figure above. The password as shown on top in Fig. 9 (above) is for the keystore. The password in the bottom line of Fig. 9 refers to the standard self-signed Tomcat certificate.
Btw.: The localhost entry has to be replaced by the correct URL.
2. – Create aCSR
%JAVA_HOME%\bin\keytool -certreq -keyalg RSA -alias tomcat -file %CATALINA_HOME%\conf\certreq.csr -keystore %CATALINA_HOME%\conf\my.keystore
We only have to enter the keystore password in the command line during this process.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC2jCCAcICAQAwZTELMAkGA1UEBhMCY2ExCzAJBgNVBAgTA1FDMREwDwYDVQQHEwhNb250cmVh
bDEQMA4GA1UEChMHc3VwcG9ydDEQMA4GA1UECxMHc3VwcG9ydDE5MBAGA1UEAxMJbG9jYWxob3N0
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvau7578dFE4imrkfF/HQtcTr1RNDY/U6
w55910Ywdrp4yTDRFXBz/GqdLdjyfPpqoCV5X10GhinVwUhZn0IKR0L8fYqNCJDElEBnMwsImU/a
awWuzV+yTqbvnpOXy/8b6wQHIQPL1x8lmkFciO+Hy7+5xOn0Axn5NbWXWUXZ64gPhWmACHEZ504j
fu+Vk4CKY/oFjr5auZlPzVYGKlwe0gnJZebZ+R5hinHI4sm44B8YYoPGjLVB1c1w8V5DZNaG+X5B
pOJDsHQ5xuQK9KCJ+A4iPA5bxBT2rtH8jh/ejr7pl2KPKesm/8ic6CYzgEgIjcMz4hc2uN7Y8v1d
3HamhQIDAQABoDAwLgYJKoZIhvcNAQkOM5EwHzAdBgNVHQ4EFgQUpdL2hMLEiVOU+XbzgMKC+Zhd
+kswDQYJKoZIhvcNAQELBQADggEBAI9yyba3U0BA5M2f/MpPY6oWCPA8PnrPfeDQr2HJT51Hxsdx
9bOF+u+4rWfiwbeyOre8CVNF+TWIG50lrUq9ss4zBUcW/u2+rWaOIIB392G3y8dNX1ODCxKGWQLB
etp5yaDxDjei4nxtLyRPnFUOXQJug0H8uZRk8+M+nYYBqGlnAlQAmVvl6j+Lj5czxzRtwuLHiHuP
y5D5mjuxc56JtIU+HUFZm0GTGdVMmq0mu8khIj7u5CY45mwEGZYdeVOycWmNb02pXuNfdP5gXm25
2iW0PzowJrxoVDavgis92LM9nUEKrFNV5rkQ9Ky8agm3/o0IuP/U7dYwj68w1dV8tK4=
-----END NEW CERTIFICATE REQUEST-----
```

Figure 10: CSR Example - Textfile

3. – Request the SSL certificate at the CA and use the CSR created in step 2
This process differs from organization to organization.
4 – Import Root and SSL certificates (chain certificates)

```
keytool -import -alias root -keystore %CATALINA_HOME%\conf\my.keystore -trustcacerts -file
<filename_of_the_root_certificate_full path)>

keytool -import -alias tomcat -keystore %CATALINA_HOME%\conf\my.keystore -file
<filename_of_the_SSL_certificate_full path)>
```

5 – Change connectors configuration in server.xml
This illustrates the basic Feedback server configuration file using a PKCS12 keystore which is decribed in the following section.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
       keystoreFile="conf/feedback.p12"
       keystoreType="PKCS12"
       keystorePass="passphrase"
       sslProtocol="TLS"
/>
```

Listing 1: Connector – server.xml

PKCS12 Keystore

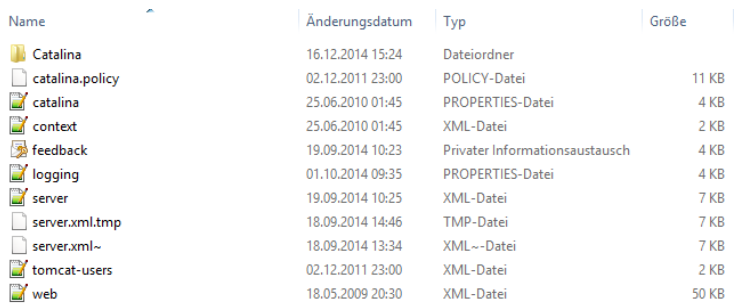
Tomcat currently operates only on JKS, PKCS11 or PKCS12 format keystores. The JKS format is Java's standard "Java KeyStore" format and is the format created by the keytool command-line utility as shown in the previous section. This tool is included in the JDK as mentioned. The PKCS12 format is an internet standard and can be manipulated via (among other things) OpenSSL[9] for example.
Each entry in a keystore is identified by an alias string. Whilst many keystore implementations treat aliases in a case insensitive manner, case sensitive implementations are available. The PKCS11 specification, for example, requires that aliases are case sensitive. To avoid issues related to the case sensitivity of aliases, it is not recommended to use aliases that differ only in case.
To import an existing certificate into a JKS keystore, please use the aforementioned way with the JDK keytool. Please note that OpenSSL often adds readable comments before the key, but keytool does not support that. So if your certificate has comments before the key data, remove them before importing the certificate with keytool.
To import an existing certificate signed by your own CA into a PKCS12 keystore (like Feedback uses in this case) using OpenSSL you would execute a command like:

```
openssl pkcs12 -export -in mycert.crt -inkey mykey.key -out mycert.p12 -name tomcat -CAfile
myCA.crt -caname root -chain
```

Tomcat can use two different implementations of SSL:

- the JSSE implementation provided as part of the Java runtime (since 1.4)
- the APR implementation, which uses the OpenSSL engine by default. (Used in the UHH DGS Feedback configuration)

The exact configuration details depend on which implementation is being used. If you configured a connector by specifying generic protocol="HTTP/1.1" then the implementation used by Tomcat is chosen automatically. In the case of DGS-Feedback in the UHH setup the APR configuration is used.
As you can see in the following figure the feedback.p12 PCKS12 keystore file is located in Tomcat's /conf folder together with the server.xml file.



Name	Änderungsdatum	Typ	Größe
Catalina	16.12.2014 15:24	Dateiordner	
catalina.policy	02.12.2011 23:00	POLICY-Datei	11 KB
catalina	25.06.2010 01:45	PROPERTIES-Datei	4 KB
context	25.06.2010 01:45	XML-Datei	2 KB
feedback	19.09.2014 10:23	Privater Informationsaustausch	4 KB
logging	01.10.2014 09:35	PROPERTIES-Datei	4 KB
server	19.09.2014 10:25	XML-Datei	7 KB
server.xml.tmp	18.09.2014 14:46	TMP-Datei	7 KB
server.xml~	18.09.2014 13:34	XML~-Datei	7 KB
tomcat-users	02.12.2011 23:00	XML-Datei	2 KB
web	18.05.2009 20:30	XML-Datei	50 KB

Figure 11: Conf-Folder

[9] https://www.openssl.org

The server.xml makes use of the feedback.p12 keystore in its configuration as shown in the following listing.
As you can see the APR configuration is used so the AprLifecycleListener has set its SSLEngine-Attribute to „on“ in Line 4 of the following XML excerpt.

```
<?xml version='1.0' encoding='utf-8'?>

<Server port="8005" shutdown="SHUTDOWN">

  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />

  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>
  <Service name="Catalina">

    <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
               keystoreFile="conf/feedback.p12"
               keystoreType="PKCS12"
               keystorePass=
               sslProtocol="TLS" />
```

Listing 2: server.xml

The configuration settings of the SSL-Connector can be seen in listing 2 above. To make use of SSL we have to enable the connector for port 8443 and set `SSLEnabled="true"` as well as `secure="true"`.
The keystore configuration is managed by the entries `keystoreFile`, `keystoreType` and `keystorePass`. The `sslProtocol` has been set to TLS.
Below you can see the non SSL configuration in the server.xml. If you like to use SSL only in your web-application please comment this section out.

```
    <Connector port="8080" protocol="HTTP/1.1"
               maxPostSize="0"
               connectionTimeout="20000"
               redirectPort="8443" />

    <Engine name="Catalina" defaultHost="localhost">

      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
             resourceName="UserDatabase"/>

      <!-- Define the default virtual host
           Note: XML Schema validation will not work with Xerces 2.2.
       -->
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true"
            xmlValidation="false" xmlNamespaceAware="false">
      </Host>
    </Engine>
  </Service>
</Server>
```

Listing 3: server.xml (http only connector)

4.2 Folder structure and deployment

In the "webapps" directory of the Tomcat server there are two separate folders deployed for the Feedback-System. These two folders have some different purposes. This means there is

- one folder for the application itself (APP-Folder)

./webapps/KorpusFeedbacksystem

- one folder for the data of the Feedback System (DATA-Folder)

./webapps/feedback/KorpusFeedbacksystem

4.2.1 APP-Folder

- Contains the application-logic of the GWT app
- Binding to the Feedback-Datastorage is carried out by ./WEB-INF/classes/ KorpusFeedbacksystem.properties

The „root-dir" is here set to `/webapps/feedback/KorpusFeedbacksystem` (XML-Data Storage) the data keeping area of the Feedback App

If the path configuration should differ in your setup you have to adapt the paths at this point

- /WEB-INF/lib →jars

In this folder you can find some additional libraries that the Feedback platform uses in order to dissolve the Java dependencies of the application (classic deployment of external libs)

- /WEB-INF/classes

This folder contains the individual programmed files from the developers of Feedback
(classic deployment of compiled java bytecode)

- The initial URL by which the Feedback application is available is mapped to the welcome file `./webapp/Korpusfeedbacksystem/KorpusFeedbacksystem.html`

Mappings i.e. for this start page can be changed in the web.xml file (welcome-file-list).
The web.xml is called deployment descriptor and provides the opportunity of configuring more parameters such as upload/download mappings, login configurations etc.

- Contains the image icons for the frontend

4.2.2 Data Folder

The folder structure for the application data is shown as follows

/packidge

Contains all packidges for all usergroups
The structure and purpose of these packidge files will be content of the following chapters. Basically, packidges are questionnaires encoded as XML data. These questionnaires are consumed by the Feedback system and processed together with the user answers. This documentation is also a guide on how to create new questionnaires and how to deal with the resulting XML in the corresponding USER-Folders.

/role

Different users can act in different roles such as administrators, employees, focus-users or standard-users.
In this directory you can find metadata related questionnaires for registration, user profile change or personal data. These are stored in the role folder so that the profiles can be defined in dependency of the roles (i.e. in different languages).
There are also naming conventions for the different roles in the application. Please take a look at the following excerpt for more details.
There are two pre-defined roles in the system:

- /role/ROLE_admin/
- /role/ROLE_standard/

Each of these role directories contain all of the system packages such as

categories.xml
→ Packidge-Reference: Which packidges are assigned to the certain user role/group.
The categories also contain the minimum score values that have to be achieved by the user to get access to the questionnaires of the certain category, individual assignments to certain packidges for the group, weights etc. See chapter 5.2 for more details on the application concept.
The assignments of packidges to user groups are carried out dependent on the achieved user-score.

help.xml
→Determines which help information has to be available for the members of this group. In the system file "help.xml" all help videos and texts are contained which are defined for the certain user group (role). The order of the items is significant here. The default sequence can be overwritten directly in the system package questionnaires (chapters 5.1 & 5.2). It is possible to arrange help videos for each questionnaire packidges individually. The administrator is able to define which video/helpsequence is shown on top etc. There will be more detailed information on metadata related questionnaires in the following chapters.

packidge.CHANGE_PROFILE.xml
→ Change profile questionnaire (package) for all users in one group
→Cf. chapter 8.3.1

packidge.PROFILE.xml
→ Personal data for pre-registered users
→ Cf. chapter 8.3.2

packidge.REGISTRATION_PROFILE.xml
→ User profile registration questionnaire
→ Only for standard-users: Data from registration process/social data

/user

The user folder contains data that can be assigned directly to the user. The files in the user directory are diverse. Starting from system xml data consumed by the Feedback system in order to handle the user as a system entity, up to multimedia data from individual comments (video, jpg) and answered questionnaires containing all the important feedback-result-information for iLex.
Have a look at the following figure for an example user directory view.

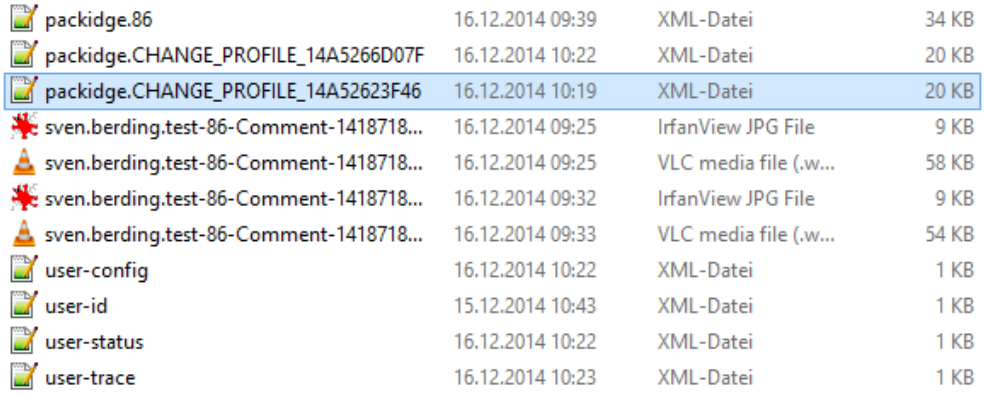


Figure 12: Directory view - USER_Sven.Berding.Test

Among the files in the figure above you can find the following initial system files to be consumed by the application:

Default system files

user-config.xml, user-id.xml, user-status.xml, user-trace.xml

Not all of the aforementioned files have to be issued by the administrator herself/himself when creating a user by hand. (cf. chapter 6.2).
Furthermore there are video COMMENT files available (.webm) including a preview image for each of the video comment streams. These comments can be:

- Questionnaire comments
- CHANGE-PROFILE COMMENTS
- Commit page comments

The file formats we are dealing with here are JPG (for the preview images) , MP4 and WEBM videos.
The deposition of change requests for the user profile are also carried out by xml files. The whole application is package-based. Under the hood these change xml request files are nothing else than a result of a metadata related questionnaire. For details on questionnaire results see chapter 9.
The answered questionnaire files (as xml) here are intended for the iLex evaluation via XSLT. iLex itself catches the data from the file system here and cuts the files from the Tomcat file system in order to ingest the packages into its database. It is also possible that users fill out questionnaires partially. The partial information is stored here in the corresponding packidge file as well. iLex does not touch partially answered packidges until the global „status“ attribute has been set to “committed”.

4.2.3 Folder permissions

Since the application stores the resulting xml data and some additional resources in the specific user folder it has to have WRITE permissions on that directory. Only READ permissions are necessary for the role and packidge directory. The latter two are only accessed to read data consumed by the application itself.

4.2.4 Deployment of new content

In order to deploy new content to the running system we have to make sure that it is not necessary to restart Tomcat. Furthermore an administrator has to deploy the content in the right order to not disturb the running system. For example, newly deployed packidges have to be available in the packidge directory before they are referenced by role or user configuration files. It has to be clear which directories an administrator does control alone and which directory the admin has to share with Tomcat.

4.3 Video / Streaming

The Feedback application utilizes “xuggler”[10] which is a free open-source library for Java developers that can be used to uncompress, manipulate, and compress recorded or live videos in real time.
This chapter gives only a short overview on the video streaming handling of the Feedback application.

4.3.1 Xuggler

“Xuggler” uses the very powerful FFmpeg media handling libraries under the hood, essentially playing the role of a java wrapper around them. It is an easy way to uncompress, modify, and re-compress any media file (or stream) from Java.

[10] www.xuggle.com/xuggler

FFmpeg[11] is a complete, cross-platform solution to record, convert and stream audio and video, supporting numerous formats. However, Xuggler's use is not limited to just providing an easy access to the complex FFmpeg native libraries.
The xuggler libraries are located in Feedback's .\webapps\feedback\KorpusFeedbacksystem\lib directory which is part of Feedbacks data folder structure.

4.3.2 Organization of streams

The videos of the feedback system are integrated into the questionnaire packages (both content related and metadata related). As we can see in the snippet below there are two alternative video files offered by the system. Depending on the browser settings and codecs of the target system either a .mp4 file or a .webm file is used for streaming.

```
<page index="0" topic="willkommen">
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270"/>
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.webm" />

        <text>
            Herzlich Willkommen im Feedback. Hier geht es jetzt darum, das Feedback-Programm näher kennenzu
            Im Anschluss kannst du dann Aufgabenpakete machen, in denen wir dich zu verschiedenen Gebärden
            &lt;br&gt; Klicke jetzt bitte auf „Weiter".
        </text>
    </content>
</page>
```

Listing 4: Position of video links in the page content of a packidge

[11] https://www.ffmpeg.org

These files are organized by an m3u8 Playlist which provides metadata information for the streaming process such as resolution, codecs, bandwidth etc.
The following listing refers to the video streams in the “welcome tutorial page” in the example listing above. This is how the metadata is provided for productional use.

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=790122,CODECS="avc1.640029",RESOLUTION=960x720
./tutorial_willkommen_1/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=36679,CODECS="avc1.640029",RESOLUTION=960x720,URI="./tutorial_willkommen
_1/iframe_index.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=508730,CODECS="avc1.640029",RESOLUTION=720x540
./tutorial_willkommen_2/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=27570,CODECS="avc1.640029",RESOLUTION=720x540,URI="./tutorial_willkommen
_2/iframe_index.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=415744,CODECS="avc1.640029",RESOLUTION=640x480
./tutorial_willkommen_3/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=22295,CODECS="avc1.640029",RESOLUTION=640x480,URI="./tutorial_willkommen
_3/iframe_index.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=272528,CODECS="avc1.640029",RESOLUTION=480x360
./tutorial_willkommen_4/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=15795,CODECS="avc1.640029",RESOLUTION=480x360,URI="./tutorial_willkommen
_4/iframe_index.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=164073,CODECS="avc1.4d401f",RESOLUTION=360x270
./tutorial_willkommen_5/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=9963,CODECS="avc1.4d401f",RESOLUTION=360x270,URI="./tutorial_willkommen_
5/iframe_index.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=162058,CODECS="avc1.42c01e",RESOLUTION=360x270
./tutorial_willkommen_6/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=9043,CODECS="avc1.42c01e",RESOLUTION=360x270,URI="./tutorial_willkommen_
6/iframe_index.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=91372,CODECS="avc1.42c01e",RESOLUTION=240x180
./tutorial_willkommen_7/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=4974,CODECS="avc1.42c01e",RESOLUTION=240x180,URI="./tutorial_willkommen
_7/iframe_index.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=36800,CODECS="avc1.42c01e",RESOLUTION=120x90
./tutorial_willkommen_8/prog_index.m3u8
#EXT-X-I-FRAME-STREAM-
INF:BANDWIDTH=1319,CODECS="avc1.42c01e",RESOLUTION=120x90,URI="./tutorial_willkommen_8
/iframe_index.m3u8"
```

Listing 5: tutorial_willkommen.m3u8 playlist

The playlist defines different parameters for different resolutions and runtime environments such as PCs with large displays or mobile devices with only small screens. As you can see above some additional .m3u8 metadata can be nested inside of one global playlist.

```
./tutorial_willkommen_7/prog_index.m3u8
#EXTM3U
#EXT-X-TARGETDURATION:20
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:20.00000,
#EXT-X-BYTERANGE:226540@0
a.ts
#EXTINF:6.32000,
#EXT-X-BYTERANGE:74072@226540
a.ts
#EXT-X-ENDLIST
```

Listing 6: prog_index.m3u8

For example, the iframe index file in the next listing is referenced in that way by the top level m3u8-playlist. The iframe index data is used for fast forward/rewind of video sequences.

```
tutorial_willkommen_7/iframe_index.m3u8
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-I-FRAMES-ONLY
#EXTINF:10.00000,
#EXT-X-BYTERANGE:6204@376
a.ts
#EXTINF:10.00000,
#EXT-X-BYTERANGE:7520@213004
a.ts
#EXTINF:5.88000,
#EXT-X-BYTERANGE:7708@410780
a.ts
#EXT-X-ENDLIST
```

Listing 7: iframe index file

5. The Application Concept

The user didactical concept of the feedback application is based on a gamification approach in order to increase the user's motivation. The goal is to keep the user's commitment and the willingness to check back to the system regularly in order to answer more questionnaires.
The application differenciates between multiple categories of questions such as for beginners, intermediates and expert level users. Depending on the user's state there are different questionnaires accessible. A beginner level user for instance is not allowed to access expert level questionnaires and so on. While the user's group membership does not change at all with additional gained scores the user's status will change with efford and therefore a higher level score.
The following chart shows the integration of this user concept inside the feedback app.

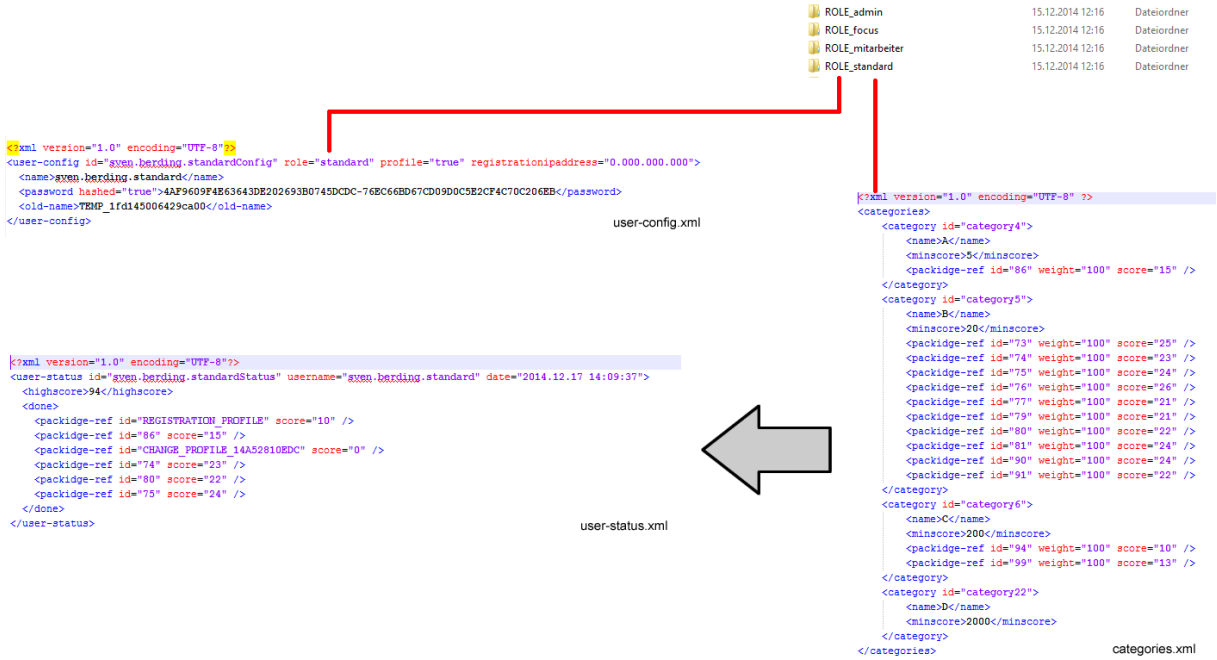


Figure 13: Integration of the user concept

5.1 Mapping of users, roles and questionnaires including the score concept

As shown in the figure above one user is associated with a role by the user-config.xml file. The role attribute of the user-config tag is set to the role name (here: standard).

```
<?xml version="1.0" encoding="UTF-8"?>
<user-config id="sven.berding.testConfig" role="standard" profile="true">
<name>vorname.nachname.standard</name>
<password>passwordUSERID</password>
</user-config>
```

Listing 8: user-config.xml

Inside the group/role itself the file “categories.xml“ determines which questionnaires the members are allowed to view and which score is nescessary to access different categories of questionnaires. (<minscore>).
The attribute „score“ contains the score a user can achieve through a fully answered questionnaire.

```
<?xml version="1.0" encoding="UTF-8" ?>
<categories>
	<category id="category4">
		<name>A</name>
		<minscore>5</minscore>
		<packidge-ref id="86" weight="100" score="15" />
	</category>
	<category id="category5">
		<name>B</name>
		<minscore>20</minscore>
		<packidge-ref id="73" weight="100" score="25" />
		<packidge-ref id="74" weight="100" score="23" />
		<packidge-ref id="75" weight="100" score="24" />
		<packidge-ref id="76" weight="100" score="26" />
		<packidge-ref id="77" weight="100" score="21" />
		<packidge-ref id="79" weight="100" score="21" />
		<packidge-ref id="80" weight="100" score="22" />
		<packidge-ref id="81" weight="100" score="24" />
		<packidge-ref id="90" weight="100" score="24" />
		<packidge-ref id="91" weight="100" score="22" />
	</category>
	<category id="category6">
		<name>C</name>
		<minscore>200</minscore>
		<packidge-ref id="94" weight="100" score="10" />
		<packidge-ref id="99" weight="100" score="13" />
	</category>
	<category id="category22">
		<name>D</name>
		<minscore>2000</minscore>
	</category>
</categories>
```

Listing 9: categories.xml

Please note: There is also a score attribute inside a questionnaire xml packidge. The score in this file can be overwritten inside the above-mentioned “category.xml” file.

5.2 Random component in a questionnaire

If there are only a few answered questionnaires of one kind available in the results the administrator is able to make an intervention on which packidges of questionnaires of a category is represented to a user with which priority level.
This is carried out by the weight attribute in the categories.xml or in the packidge file itself.
By the weight attribute administrators are able to control which packidge gets a higher propability to be the next one that is represented to the user. This way concurrency between different packidges can be driven.
This is how an administrator has control over the components so that she/he can determine which contents to present in a certain moment to a certain user. The first way to take this control is the already discussed “weight” parameter. The second is the “score” attribute parameter that also has already been refered to in this chapter.
These two parameters have in common, that they are defined in a package and can be overwritten in the categories.xml file for a group or in the user-category.xml file (user folder) for a certain user if a personal packidge is supposed to be referenced (see chapter 5.3).
The reference to a certain packidge is implemented by the <packidge-ref> xml tag. The id attribute matches with a questionnaire with the corresponding id.
The status of a user is implemented in xml as well. As soon as a user has completed a packidge (which does not nescessarily has to be a content related questionnaire but also a change profile packidge for instace) an entry inside of the user-status.xml is made. The user-staus.xml is located

inside the user's personal folder. As you can see above the user scores are summarized inside the file in order to determine the user's current total score which is nothing else than the user status itself. As you can also see above, completing metadata related questionnaires does not result in further scores (CHANGE_PROFILE packidge score = 0).
But this is also configurable if desired. The packidges „registration", „collection of social data" and „change of social data" contain their scores directly inside their xml. If you want to give the user points for handling these packidges the scores have to be configured directly in their own xml or have to be overwritten in "categories.xml" or "user-categories.xml" in case of personal packidges.
The standard value of weight and score is 0, if one attribute is missing.

5.3 Personal questionnaires / packidges

As mentioned earlier an administrator has the opportunity to map certain questionnaires to certain groups / roles. The users being members of a group then will have access to all these pre-defined packidges.
But administrators are also able to map the questionnaire items directly to one user without considering a specific role that is to say a specific group membership. In this case the user is directly associated with a packidge.xml by the "user-categories.xml" file. This file has to reside inside the user's specific directory.
This feature has been implemented in order to become able to make further specific inquiries to users while referring to a special questionnaire. In that way the inquiries can be easily targeted to a certain user and questionnaire combination.
As discussed the only technical difference an administrator has to be aware of is that the reference to a package is made in the user's own folder here and not in the role folder.
The packidge-reference is located as a separate file inside of the user directory.

```
<?xml version="1.0" encoding="UTF-8" ?>
<user-category id='myCategory'>
       <packidge-ref id='MP000001' weight='100' score='21'/>
</user-category>
```

Listing 10: user-category.xml

A further example:

```
<?xml version="1.0" encoding="UTF-8"?>
<user-category id="Für mich">
       <packidge-ref id="84" weight="100" score="6" />
       <packidge-ref id="084" weight="100" score="6" />
</user-category>
```

Listing 11: user-category.xml

The screenshot below shows an unlocked category for a user in the the web GUI frontend. An unloked category item is clickable and therefore accessible for a user due to the <minscore> tag in category.xml.

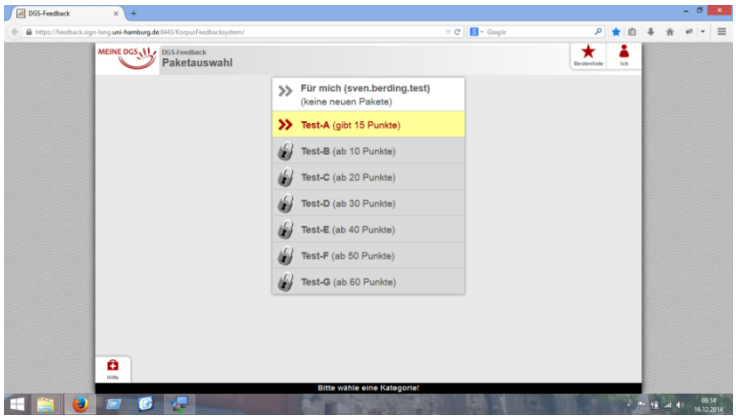


Figure 14: Categories in the GUI

As mentioned before the user status is summarized in a central xml file called “user-status.xml”. The summarized score becomes visible in the frontent GUI as shown below.

Figure 15: Highscore list after completing packidge74.xml

After exeedance of the minscore barrier, a new category becomes available and unlocked.

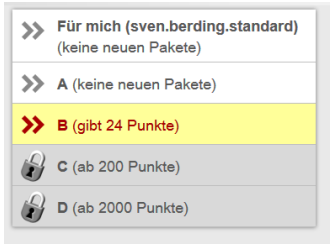


Figure 16: Category B becomes unlocked after Cat. A

Since category B contains multiple packidges in this example there is no access to a next category after completing the initial questionnaire. the 24-points-questionnaire in fig. 16 and the 21-point-questionnaire below in fig. 17. Both are in the context of the same category.

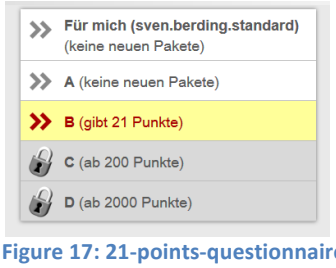


Figure 17: 21-points-questionnaire / still category B

Figure 18 resumes the score/status concept of the Feedback application. The aforementioned concept becomes visually integrated by this overview in the web GUI.

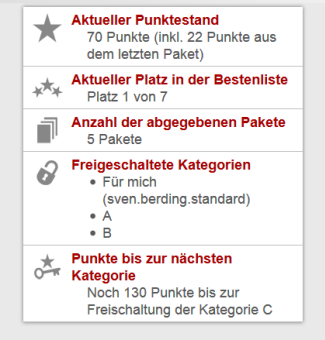


Figure 18: Score and status concept

As a part of the application concept the missing points to achieve a new category are listed in this screen and therefore reflect the entities and concepts presented in the xml dialect during this chapter.

6. Creating users and roles

There are different user types available in the feedback system. Users can act in the standard role by the default registration process from the Feedback website. In this case the user does not have to be created manually by the IT staff. In case of other users than in the standard role such as administrators or iLex users it might be necessary to create them from scratch manually.

6.1 Different types of users

Technically, the only user role in need for the system is the standard user. This kind of user is added to the system by the default registration process in the web frontend. It is also the only type of user that has a "REGISTRATION_PROFILE.xml" file in its user directory after registration. This profile information is intended for ingestion into the iLex database, and since every new registered user is mapped automatically to the standard role the REGISTRATION_PROFILE is uniquely present in the standard user folders.
A special role is intended for the admin user. In order to enable an administrator to test a new installation of the application it is necessary that packidges are never stored in the system. Otherwise packidges would be only available for one single answering process of the administrator. This role is furthermore the only way to reference certain packidges by parameter directly from the browser URL such as "https://meinedgs.sign-lang.uni-hamburg.de:8443/KorpusFeedbacksystem/?packidge=92" for example. Users not of the status of an administrator are not able to use URL parameters. The parameters will be skipped in this case.

6.2 Create users and roles

Users

User and roles can be created manually in addition to the default user registration process.
The only file you need to create a user in the file system from scratch is the "user-config.xml". In contrast, the "user-trace.xml" is created automatically by the system after the first login of the new user. The trace file contains the information of how far the user is progressed by answering a questionnaire. If an answering process is interrupted the system itself is able to resume the process when the user likes to continue with the questionnaire.
Stored information: which packidge file, exact position in the corresponding packidge.
The file "user-id.xml" has also not to be created manually because the file is only present in the case that the data sets originating from iLex.
In case of personal packidges the administrator has to create a "user-category.xml" file in the way described.
If you like to create Feedback-Users manually you have to follow these following steps.
Navigate to

\webapps\feedback\KorpusFeedbacksystem

and you will find the following folder structure

Figure 19: Basic data directory

Navigate to the user directory. Here you have to create a new directory for the new user.
Please use the following name convention for the folder: USER_firstname.lastname.rolename

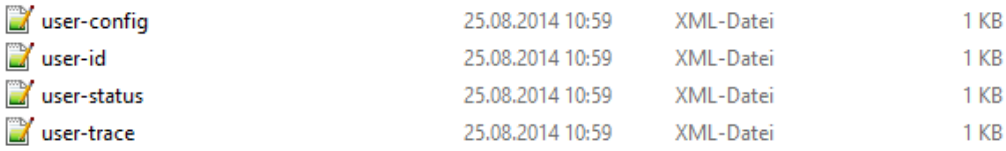


Figure 20: Initially created xml data inside the user folder

Now you have to manually create two files (in case of users originating from iLex). One is "user-config.xml".

Create the "user-config.xml" according to the following template

```
<?xml version="1.0" encoding="UTF-8"?>
<user-config id="firstname.lastname.rolenameConfig" role="rolename" profile="true">
    <name>firstname.lastname.rolename</name>
    <password>passwordUSER-ID</password>
</user-config>
```

Listing 12: user-config.xml

In case of a user originating from iLex the second file you have to initially create is "user-id.xml".
The initial password is a string combined with the USER ID integer value. After the first login of the user the password becomes RSA encrypted/hashed. Passwords of standard users are hashed immediately on registration.
The "user-id.xml" is also used for the password recovery process in case a user requests a new one. Thereby the hashed password will be replaced by a clear text string.

```
<?xml version="1.0" encoding="UTF-8"?>
<user-id>Integer</user-id>
```

Listing 13: user-id.xml

The integer value in the listing above is a simple user id such as an integer value of 90.
This is the initial content for the "user-status.xml" (empty tag). This file does not have to be created by administrators manually. As we have already discussed the status of a user is implemented in xml as well. As soon as a user has completed a packidge (which does not nescessarily has to be a content related questionnaire but also a change profile packidge for instace) an entry inside of the "user-status.xml" is made. The user scores are summarized inside the file in order to determine the user's current total score which is nothing else than the user status itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<user-status id="firstname.lastname.rolenameStatus" username="firstname.lastname.rolename" date="2014.02.16 16:40:20">
</user-status>
```

Listing 14: user-status.xml

The "user-trace.xml" will be created automatically by the application after the first login of a user. It stores information about the answering process of questionnaires in order to resume aborted processes and to locate the point where a user stopped answering a packidge.

```
<?xml version="1.0" encoding="UTF-8"?>
<user-trace id="firstname.lastname.rolenameTrace" username="firstname.lastname.rolename" date="2014.02.16 16:40:20" />
```

Listing 15: user-trace.xml

Roles

Roles can be created in the file system according to the same paradigm utilized with user creation. The naming convention as described by “ROLE_rolename” has to be kept. Since the role name itself follows an underscore token (_) the role name can be easily referenced from the system xml.
If you also want to create new roles for the system users you have to add a folder to the role directory.

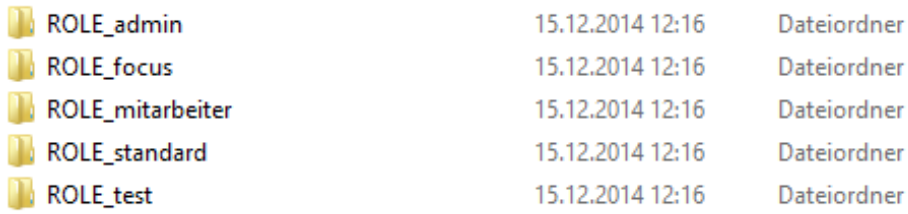


Figure 21: ROLE directory folder structure

The naming of the role directories follow the naming convention ROLE_rolename and do initially contain two files.

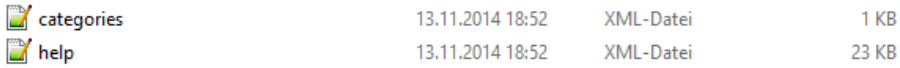


Figure 22: Initial content of the role directory

As we have seen before, the file “categories.xml” has to be configured as decribed in the previous chapter.

```
<?xml version="1.0" encoding="UTF-8" ?>
<categories>
    <category id="category7">
        <name>Mitarbeiter-A</name>
        <minscore>5</minscore>
        <packidge-ref id="73" weight="100" score="25" />
        <packidge-ref id="79" weight="100" score="21" />
        <packidge-ref id="86" weight="100" score="15" />
    </category>
    <category id="category20">
        <name>Mitarbeiter-B</name>
        <minscore>20</minscore>
        <packidge-ref id="74" weight="100" score="23" />
        <packidge-ref id="75" weight="100" score="24" />
        <packidge-ref id="76" weight="100" score="26" />
        <packidge-ref id="77" weight="100" score="21" />
        <packidge-ref id="80" weight="100" score="22" />
        <packidge-ref id="81" weight="100" score="24" />
        <packidge-ref id="90" weight="100" score="24" />
        <packidge-ref id="91" weight="100" score="22" />
    </category>
    <category id="category21">
        <name>Mitarbeiter-C</name>
        <minscore>200</minscore>
        <packidge-ref id="94" weight="100" score="10" />
        <packidge-ref id="99" weight="100" score="13" />
    </category>
    <category id="category23">
        <name>Mitarbeiter-D</name>
        <minscore>2000</minscore>
    </category>
</categories>
```

Listing 16: categories.xml

Please note:
The scores inside the “categories.xml” overwrite the scores inside the questionnaire packidges. Questionnaires are linked here too in the way already decribed in chapter 4.2.2.

Creation of help files for a new role/group

As you can see in the listing below the help.xml file consists of one or many content blocks.

```
<?xml version="1.0" encoding="UTF-8"?>

<help>
    <content id="goldener pfad" type="multimedia" hratio="360" vratio="270">
    <content id="text gebaerde" type="multimedia" hratio="360" vratio="270">
    <content id="kategorien" type="multimedia" hratio="360" vratio="270">
    <content id="punkte" type="multimedia" hratio="360" vratio="270">
    <content id="abgeben" type="multimedia" hratio="360" vratio="270">
    <content id="antwortknoepfe" type="multimedia" hratio="360" vratio="270">
    <content id="uebersicht" type="multimedia" hratio="360" vratio="270">
    <content id="persoenliche daten" type="multimedia" hratio="360" vratio="270">
    <content id="videoupload" type="multimedia" hratio="360" vratio="270">
    <content id="kamera" type="multimedia" hratio="360" vratio="270">
    <content id="kamera_firefox" type="multimedia" hratio="360" vratio="270">
    <content id="kamera_chrome" type="multimedia" hratio="360" vratio="270">
    <content id="kommentar" type="multimedia" hratio="360" vratio="270">
    <content id="listen" type="multimedia" hratio="360" vratio="270">
    <content id="nubsileiste" type="multimedia" hratio="360" vratio="270">
    <content id="abmelden" type="multimedia" hratio="360" vratio="270">
    <content id="benutzername" type="multimedia" hratio="360" vratio="270">
    <content id="textvalidierung" type="multimedia" hratio="360" vratio="270">
    <content id="mail" type="multimedia" hratio="360" vratio="270">
</help>
```

Listing 17: content blocks in the help file

Each content block contains a sructure according to the xml excerpt below.

```
<content id="punkte" type="multimedia" hratio="360" vratio="270">

    <image
        alt="Standbild aus dem Gebärdenvideo"
        src="https://feedback.sign-lang.uni-hamburg.de/hilfeseiten/hilfe_punkte.jpg"
        srcset="https://feedback.sign-lang.uni-hamburg.de/hilfeseiten/hilfe_punkte_2x.jpg 2x"
    />
    <video src="https://feedback.sign-lang.uni-hamburg.de/hilfeseiten/hilfe_punkte.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/hilfeseiten/hilfe_punkte.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/hilfeseiten/hilfe_punkte.webm" />

    <text>
        &lt;b&gt;Punkte und Bestenliste: &lt;/b&gt; &lt;br&gt;Für jedes vollständige Aufgabenpaket
        bekommst du Punkte. Die Höhe der Punktzahl richtet sich nicht nach dem Inhalt deiner Antworten.
        Ob du z.B. eine Gebärde kennst oder nicht kennst, ist für uns gleich wichtig. In der Paketauswahl
        kannst du schon vorher sehen, wieviele Punkte du bekommen wirst. Sie werden dir gutgeschrieben,
        wenn du ein Aufgabenpaket fertig gemacht und abgeschickt hast. In der Bestenliste siehst du,
        wieviele Punkte dazu gekommen sind und wie hoch du im Vergleich zu den anderen Teilnehmenden
        gekommen bist. Weitere Informationen findest du auf der „Ich"-Seite, z.B. wie viele Pakete du schon
        abgegeben hast und welche Aufgaben-Kategorien schon für dich freigegeben sind.
    </text>
</content>
```

Listing 18: content block structure

The single content items can be selectively integrated inside the packidges (both content-related and metadata-related) by the <help> tag.

7. Basic structure of questionnaires

General note: As already discussed in chapter 2 the term “packidge” instead of “package” is continuously used in the whole application and therefore in this present documentation. The reason for using the packidge-term is that the application is java-based. Since “package” is a reserved keyword in Java it is not possible to use the term in another context as well.
There are basically two different types of questionnaires

1. Content-Related Questionnaires
2. Metadata-Related Questionnaires

Content-Related Questionnaires

The content-related questionnaires as considered in chapter 8 are automatically generated from the iLex transcription environment. They represent the core type of questionnaires containing project-relevant questions on signs and sign uses. The basic structure and layout of questionnaires is explained by content-related- questionnaires-examples in the current chapter for this type being the most commonly used, too.

Distinction of content related questionnaires

This distinction is not of technical nature but of organizational character. There are three types of questionnaires.

Type 1: Form and meaning of signs
Type 2a: The spectrum of possible characteristics based on a meaning of a sign
Type 2b: Same as 2a plus a grouping of signs in case there are so many meanings to a special sign that the signs have to be organized once more.

Metadata-Related Questionnaires

The metadata-related questionnaires do not deal with sign contents. Instead, they handle profile- and registration master file data. Since the whole data handling and persistence structure of the application is based on a file system store paradigm every change in the user data has to be communicated to the iLex database by an xml package file. In order to meet this objective it is necessary to map the process of changing the user’s metadata on the questionnaire approach.
This becomes obvious by regarding the metadata-related questionnaires in chapter 8.3. These are stored in the role folder so that the profiles can be defined in dependency of the roles (i.e. in different languages).

packidge. PROFILE.xml (User profile – personal data)
packidge.CHANGE_PROFILE.xml (User profile – change)
packidge.REGISTRATION_PROFILE.xml (User profile – registration

A third type of questionnaires is represented by the INPUT-packidge and packidge M0000012 (Validation of lists, scale questions etc.). These are hand-made during the testing process of the application and are available for administration purposes. By means of these files it is possible to present some additional features that can be implemented into the feedback questionnaires in general such as additional types of questions containing text-verification models (REGEX). Compare to chapter 8.4.

7.1 Structure of a packidge / XML DOM template

Each questionnaire packidge follows this basic template structure. This structure becomes refined step by step in the progress of this documentation.

```
<packidge id="$id" score="$score" weight="$weight">
```

```
  <topic>Package</topic>
  <name>$name</name>
  Welcome
  Task
  (if applicable: further pages which occur in every package of this type)
  $pages (1…n)
  (if applicable: further pages which occur in every package of this type)
  commit-page
  revise-page
  retract-page
  help
</packidge>
```

Listing 19: Packidge structure skeleton

7.2 Structure of a single page as insert for $pages in 7.1

The $pages-construct in chapter 7.1 can be replaced modularly by the following page-structure-skeleton.

```
<page id="$id" index="$pagenumber" topic="$name">
  comment
  Frage/Oberste Zeile jeder Inhaltsseite
  $rows
  (ggf. letzte Zeile einer Seite, die auf jeder Inhaltsseite vorkommt)
</page>
```

Listing 20: Page structure skeleton

7.3 Combining the components for the Feedback system

If you combine the packidge element above with the corresponding page elements this will result in the following basic questionnaire structure that we will explore in the following steps. After an example tutorial in chapter 8.2 you will be able to create your own content related questionnaires for Feedback.

```
<?xml version="1.0" encoding="UTF-8"?>
<packidge id="86" score="15" weight="100">
    <topic>Paket</topic>
    <name>Einführung in das Programm</name>
    <page index="0" topic="willkommen">
    <page index="0_1" topic="dgs_text">
    <page index="0_2" topic="text">
    <page index="1" topic="navigation">
    <page index="2" topic="frage_antworttypen">
    <page index="3" topic="hilfe">
    <page index="4_1" topic="kommentar_text">
    <page index="4_2" topic="kommentar_video">
    <page index="5_1" topic="statusleiste">
    <page index="5_2" topic="sonstiges">
    <page index="6" topic="abgeben">
    <commit-page>
    <revise-page>
    <retract-page>
    <help>
</packidge>
```

Listing 21: Basic questionnaire structure

7.4 Structuring contents (Pages & rows as content of a questionnaire)

The listing above only represents a skeleton for a questionnaire consumed by the application. In the next step each page has to be filled up with content.

7.4.1 Pages

Content is organized in each page of a packidge.

```
<page index="0" topic="willkommen">
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270"/>
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image alt="Standbild aus dem Gebärdenvideo" src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.j
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_willkommen.webm" />

        <text>Herzlich Willkommen im Feedback. Hier geht es jetzt darum, das Feedback-Programm näher kennenzulernen. Im Anschluss
    </content>
</page>
```

Listing 22: Simple welcome page (topic="willkommen")

Later we will see how content blocks are presented to the user in the frontend.
The **page** construct can be skimmed through by clicking the „Weiter" button in a questionnaire. As we can see here the page above is followed by a further page with a simple identical structure. In this case the next page contains an introductory text / video.

```
<page index="0_1" topic="dgs_text">
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270"/>
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image alt="Standbild aus dem Gebärdenvideo" src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_dgs_text.jpg" s
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_dgs_text.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_dgs_text.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/tutorial/tutorial_dgs_text.webm" />

        <text>Alle Aufgaben und Erklärungen kannst du dir in DGS oder Deutsch anschauen. Mit dem Knopf hier links oben kannst du umsc
    </content>
</page>
```

Listing 23: Page containing an introductory text / video

7.4.2 Content–Blocks and rows

Contents are structured in the questionnaire. That is to say one can prepare the contents visually for the user in a way that text components are placed besides images or contents are placed one below the other.
Therefore a kind of container for contents is defined which is exactly the mentioned <content> xml construct. Inside of this tag we can place our Feedback content we like to present to the user.
If contents are supposed to be faded in from the bottom of the page we have to put these content-containers (<content>) inside the <row> container construct (Conditional Sub-Questions). Multiple <content>-elements can be placed inside the <row> construct in order to place them parallel in one row in the UI-Frontend. All that becomes more clear in the example of chapter 8.2.
A <row> with index="1" will always be shown directly in the second line which is located below the header line. Compare Fig. 23 below.
This row with index 1 acts as an initial start point for further rows of one page. The rows can be faded in from the bottom sequentially. Comp. Listing 24 below and the Fig. 23 as mentioned.

```
<page index="2" topic="frage_antworttypen">
    <comment>
    <content type="multimedia" hratio="360" vratio="270">
    <content type="multimedia" hratio="360" vratio="270">
        <image src="https://feedback.sign-lang.uni-hamburg.de/graphics/tutorial_antworten.jpg"  srcset="https:
    </content>

    <row index="1" type="buttons">

    <row id="4458" type="buttons">
    <row id="4455" type="buttons">
</page>
```

Listing 24: row index=1

<content> as well as <row> are part of a single <page>.
Pages have indexes as well so that the system is aware of the current state of the answering process of the user questionnaire.

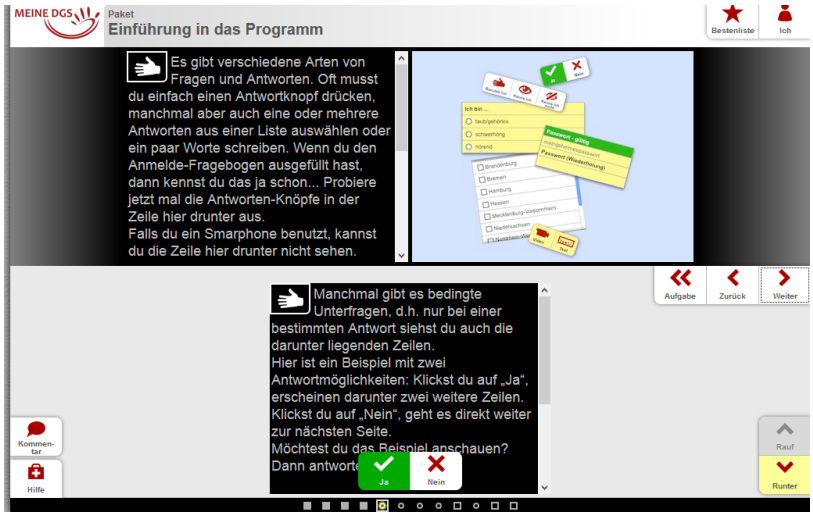


Figure 23: Second row – below the header (row index="1")

7.5 Jumping on the next page / line / conditional sub-answers

A button based navigation is most commonly used throughout the application to skim on next pages or rows. We will se this in action later on. For the goto attribute there are valid values as follows:

- next-sub-page → Default case: go to next page
- exit-page → Jump on the next page
- exit-packidge → Jump out of the packidge = Force packidge to abort
- <index> → Index of a concrete line → Lines are enumerated for that. (starting at 0);

Jump over pages

A page has to be viewed at least once. The attribute processed="true" in the page tag makes it possible to jump over pages (i.e. for the change packidge). This documentation addresses this topic in chapter 8 (Answered questionnaires as results).

8. Creation of questionnaires

In order to create your own questionnaires you have not only to be aware of the basic structure of the XML DOM of a packidge file. It is rather important to know how changes in the questionnaire xml structure are reflected directly in the GUI.
According to that the following chapter uses screenshots, xml snippets and explanations combined in order to make clear how changes in the xml cause effects in the web GUI in detail. This is carried out by an example of a content-related questionnaire as this is the most commonly used xml packidge in the application.

8.1 Preparation

In order to create your own questionnaires it is a good idea to use a text editor that allows you to define a certain computer language for the current document. The great benefit is the syntax coloration feature which allows the editor to become exactly aware of the semantics of an xml document.
The XML-Editor Notepad++[12] is an exellent freeware tool for that purpose. There are some language and syntax coloration features available. With the XML-Tools-Extension you can enable a pretty print formatting for the whole document as well.
Be also aware to check the validity of the created document so that the application does not throw any parsing error by processing the xml.

8.2 Example design (Form und Bedeutung Package)

In this chapter a questionnaire implementation for packidge75 is discussed in detail in order to show synchronisation between the programmatic aspects in the file itself and the GUI on the other side.
The xml file is part of the package “Form und Bedeutung“ which also contais further xml questionnaires.
As you can see in the listing below the membership of the questionnaire in the category B is determined by the “categories.xml” file.

[12] http://notepad-plus-plus.org

```
<?xml version="1.0" encoding="UTF-8" ?>
<categories>
 <category id="category4">
  <name>A</name>
  <minscore>5</minscore>
  <packidge-ref id="86" weight="100" score="15" />
 </category>
 <category id="category5">
  <name>B</name>
  <minscore>20</minscore>
  <packidge-ref id="73" weight="100" score="25" />
  <packidge-ref id="74" weight="100" score="23" />
  <packidge-ref id="75" weight="100" score="24" />
  <packidge-ref id="76" weight="100" score="26" />
  <packidge-ref id="77" weight="100" score="21" />
  <packidge-ref id="79" weight="100" score="21" />
  <packidge-ref id="80" weight="100" score="22" />
  <packidge-ref id="81" weight="100" score="24" />
  <packidge-ref id="90" weight="100" score="24" />
  <packidge-ref id="91" weight="100" score="22" />
 </category>
 <category id="category6">
  <name>C</name>
  <minscore>200</minscore>
  <packidge-ref id="94" weight="100" score="10" />
  <packidge-ref id="99" weight="100" score="13" />
 </category>
 <category id="category22">
  <name>D</name>
  <minscore>2000</minscore>
 </category>
</categories>
```

Listing 25: Membership in a certain category

First of all let's have a look at the basic hierarchical structure of the entire file.

```
<packidge id="75" score="24" weight="100">
    <topic>Paket</topic>
    <name>Form und Bedeutung</name>
    <page index="0" topic="willkommen">
    <page index="1" topic="aufgabe 1">
    <page id="1183" index="1" topic="123: TIER4-$SAM">
    <page id="1176" index="2" topic="115: HEIRATEN1-$SAM">
    <page id="1174" index="3" topic="115: HEIRATEN1-$SAM (HEIRATEN4-$SAM)">
    <page id="1175" index="4" topic="115: HEIRATEN1-$SAM (HEIRATEN5-$SAM)">
    <page id="1178" index="5" topic="117: RING1A-$SAM">
    <page id="1177" index="6" topic="117: RING1A-$SAM (HEIRATEN3-$SAM)">
    <page id="1182" index="7" topic="122: FAMILIE1-$SAM">
    <page id="1189" index="8" topic="132: TIER3-$SAM">
    <page id="1191" index="9" topic="133: TIER1A-$SAM">
    <page id="1190" index="10" topic="133: TIER1A-$SAM (TIER1B-$SAM)">

    <commit-page>
    <revise-page>
    <retract-page>
    <help>
</packidge>
```

Listing 26: Hierarchical structure of a sample packidge

As you can see there are some attributes inside the basic tags that specify the questionnaire handling for the system itself. The attributes are i.e. page id, index and the topic string. They also represent some additional information for the latter evaluation in iLex.
Taking a look at the first screen of the package we can see the basic structure in the GUI. In the upper part we can find the initial content that contains a video in .webm or .mp4 format.

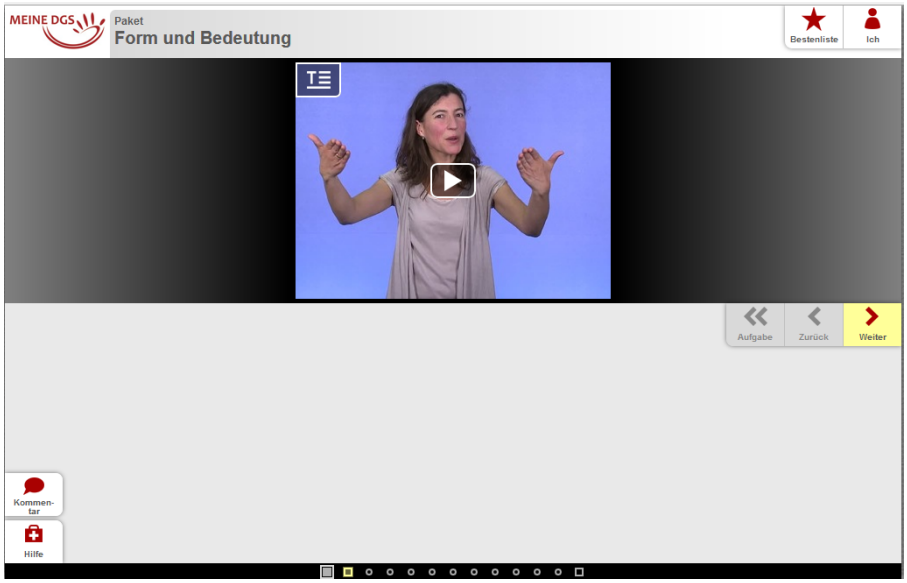


Figure 24: Screen 1 | Video Frame

In order to start with simple content and to finish with more complex structures let us first take a look at the alternative layout for the screen.
Here an alternative layout is shown. The text inside the video frame is an equivalent to the video contents in the screen above.

Figure 25: Screen 1b | Text inside a video frame

In order to show how this simple first screen corresponds to the xml content of the packidge file please have a look at the XML-Listing below (Listing 27) which is exactly the same content as shown in the screens above but this time in the xml dialect.
As you can see this is the page with index=0 - in this case the welcome page - that does not contain any content in the bottom part of the browser window.
The basic structure shows that there is only one content block that contains the text and video information.
First of all the initial start image is placed in the content block as jpg file. The video tag contains the aforementioned playlist that refers to the embedded video files inside the <video> tags. Here we can see how the video in screen 1 and 1b (figure 24 & 25) is placed inside the xml content. Two alternative video files are available depending on the user's setup.

```
<page index="0" topic="willkommen">
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270"/>
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/willkommen_aufgabentyp1.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/willkommen_aufgabentyp1_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/willkommen_aufgabentyp1.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/willkommen_aufgabentyp1.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/willkommen_aufgabentyp1.webm" />

        <text>
            Herzlich willkommen beim DGS-Korpus-Feedback. Schön, dass du mitmachst! Was erwartet dich hier?
            Wir stellen Fragen zu einzelnen Gebärdenformen und ihren Bedeutungen sowie Fragen zur regionalen
            Verbreitung von Gebärden. Viel Spaß!
        </text>
    </content>
</page>
```

Listing 27: XML for Screen 1, 1b

Referring to screen 1b (figure 25) we can see the text contents in the xml file between the <text></text> construct. If you like to place your own text content this is the right place for text strings.
Since this is the first structure to be described here you have to be aware that this one is also the simplest one. Having a look at the next screen we will find some more complex content in the second step.
In screen 2 (figure 26) there is no content in the bottom of the page as well but as we can see there are two video files placed side by side in the header area.

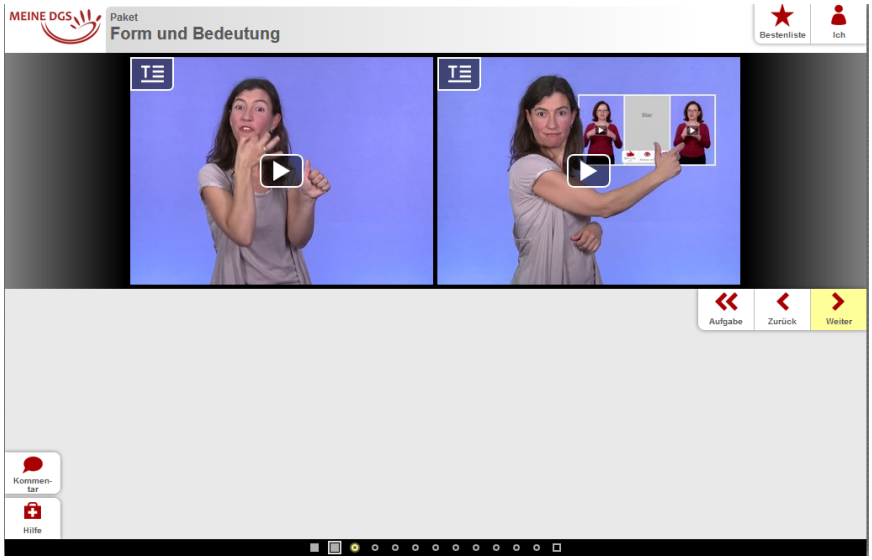


Figure 26: Screen 2 | Two video files in the header area

Although there are two video frames placed side by side, they can be easily switched to text mode as we can see here.

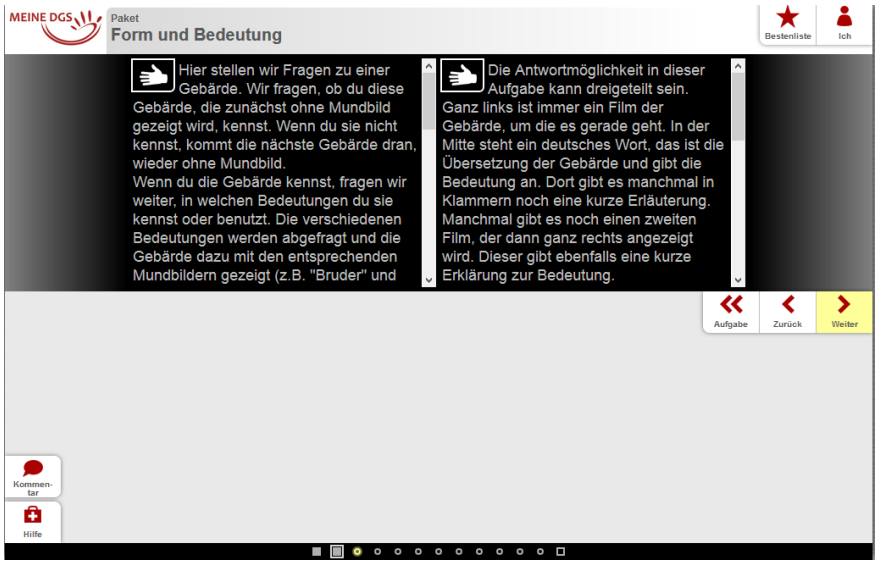


Figure 27: Screen 2 | Text mode of side by side content

The corresponding xml snipped shows what has happened in the code here. While the basic structure remains the same we can see that there are now two content blocks integrated in the structure. This causes the side by side presentation of the contents in the GUI.

```
<page index="1" topic="aufgabe 1">
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270"/>
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/aufgabentext_aufgabentyp1.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/aufgabentext_aufgabentyp1_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/aufgabentext_aufgabentyp1.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/aufgabentext_aufgabentyp1.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/aufgabentext_aufgabentyp1.webm" />

        <text>
            Hier stellen wir Fragen zu einer Gebärde. Wir fragen, ob du diese Gebärde, die zunächst ohne Mundbild
            gezeigt wird, kennst. Wenn du sie nicht kennst, kommt die nächste Gebärde dran, wieder ohne Mundbild.&lt;br&gt;
            Wenn du die Gebärde kennst, fragen wir weiter, in welchen Bedeutungen du sie kennst oder benutzt. Die verschiedenen
            Bedeutungen werden abgefragt und die Gebärde dazu mit den entsprechenden Mundbildern gezeigt (z.B. "Bruder" und
            ..............
        </text>
    </content>
    <content index="1" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/erklaerung_3teilscreen.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/erklaerung_3teilscreen_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/erklaerung_3teilscreen.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/erklaerung_3teilscreen.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/erklaerung_3teilscreen.webm" />

        <text>
            Die Antwortmöglichkeit in dieser Aufgabe kann dreigeteilt sein. Ganz links ist immer ein Film der Gebärde,
            um die es gerade geht. In der Mitte.....
            .....
        </text>
    </content>
</page>
```

Listing 28: Screen 2 XML / 2 content blocks (contents are presented side by side in the frontend)

Following the path to more complexity we regard the third screen example in the next step. This is the first time that there is also content in the bottom area of the browser GUI. This is operated by the row tag in the xml construct.

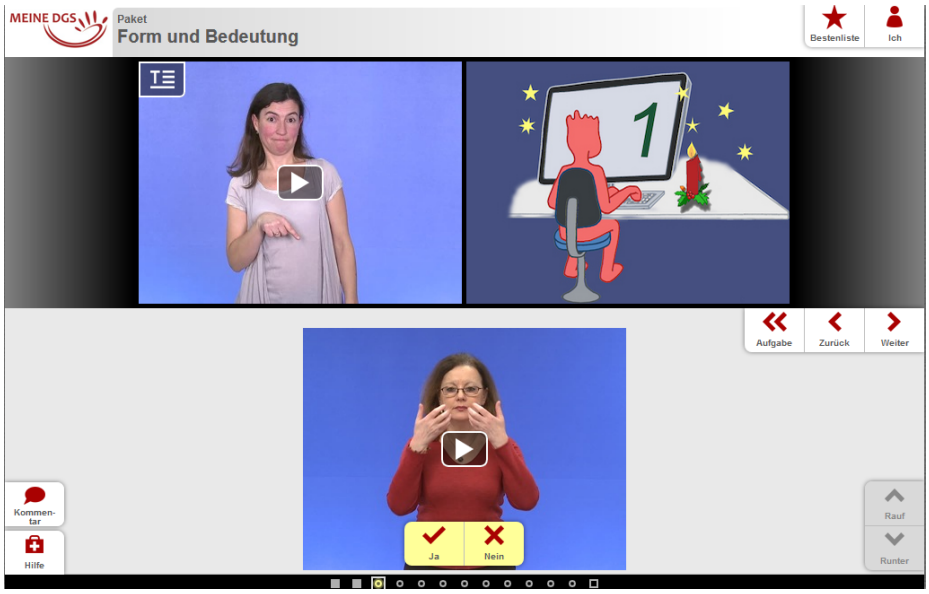


Figure 28: Screen 3 | Row content at the bottom

The index level of 0 is responsible for placing the content right beneath the header line of the browser GUI content. Alternatively it is still possible to switch to text mode as mentioned.
In the header area of the page the content is still organized as decribed by the content tags. There are two video/text frames side by side. What's new here is the row content part at the bottom of the following listing.

```
<page id="1183" index="1" topic="123: TIER4-$SAM">
    <comment>
        <content type="multimedia" hratio="360" vratio="270" />
    </comment>
    <content type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.webm" />

        <text>Kennst du diese Gebärde?</text>
    </content>
    <content type="multimedia" hratio="360" vratio="270">
        <image
            src="https://feedback.sign-lang.uni-hamburg.de/graphics/icon_figur_1.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/graphics/icon_figur_1_2x.jpg 2x"
        />
    </content>
    <row id="4066" index="0" type="buttons">
        <content id="4066" index="1" type="multimedia" hratio="360" vratio="270">

            <image
                alt="Standbild aus dem Gebärdenvideo"
                src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.jpg"
                srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378_2x.jpg 2x"
            />
            <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.m3u8" />
            <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.mp4" />
            <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.webm" />

        </content>
        <button id="4066-1" index="0" type="option" icon="YES">Ja</button>
        <button id="4066-0" index="1" type="option" icon="NO" goto="exit-page">Nein</button>
    </row>
```

Listing 29: Screen 3 XML | Row content

The row content contains the information shown at the screen bottom as described. The buttons inside the row are integrated inside the video frame itself. Do not mix it up with the buttons labeled with „Weiter“. These are not here in the row tag!
By clicking the „Runter“ button in the GUI, new row content becomes available. This is faded in from the bottom. Please be aware that we are still on the same page (<page>) when new row content becomes scrolled in as you can see in the next screen.

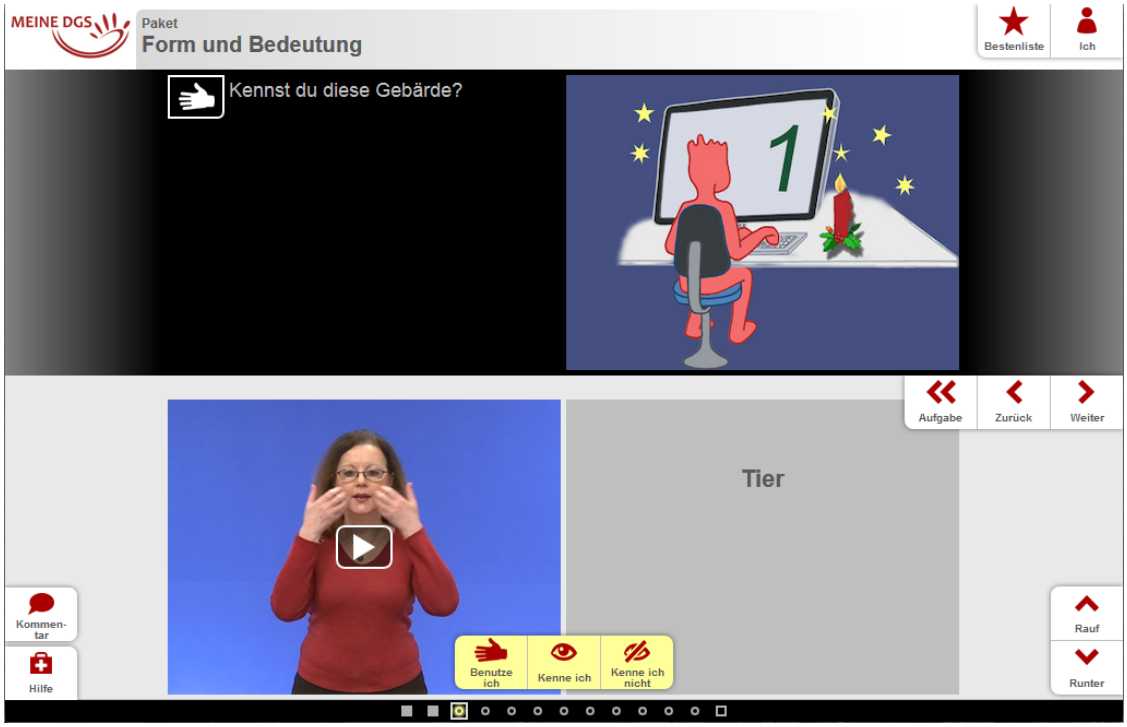


Figure 29: Screen 3 row index=2 | New row content

Please note that the row index is now not zero anymore. This (zero) is only for the initial row content. The index increases with new rows that become available. Also note that we have one more content area in the row here. There is a video and a text frame side by side inside the row content. This is achieved in the same way as already decribed by the page header area content.
Considering the button tags we are able to provide each button with a special image icon.

```
<row id="4067" type="buttons">
    <content id="4067" index="2" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.webm" />

    </content>
    <content index="1" type="keyword" hratio="360" vratio="270">Tier</content>

    <button id="4067-2" index="0" type="option" icon="SIGN_USED">Benutze ich</button>
    <button id="4067-1" index="1" type="option" icon="SIGN_KNOWN">Kenne ich</button>
    <button id="4067-0" index="2" type="option" icon="SIGN_UNKNOWN">Kenne ich nicht</button>
</row>
```

Listing 30: 2 Content areas inside of the row | Side by side content

This is specified by the icon attribute and the values SIGN_USED, SIGN_KNOWN and SIGN_UNKNOWN. Each button has to get a unique id according to the id naming convention **ROW_ID**-BUTTON_INDEX_STARTING_AT_0 (i.e. 4067-0).
At the end of a page that contains multiple rows the user is asked for some additional meanings of a sign. The user has the opportunity either not to answer at all or to answer by text or video as presented in the next Fig. 30.

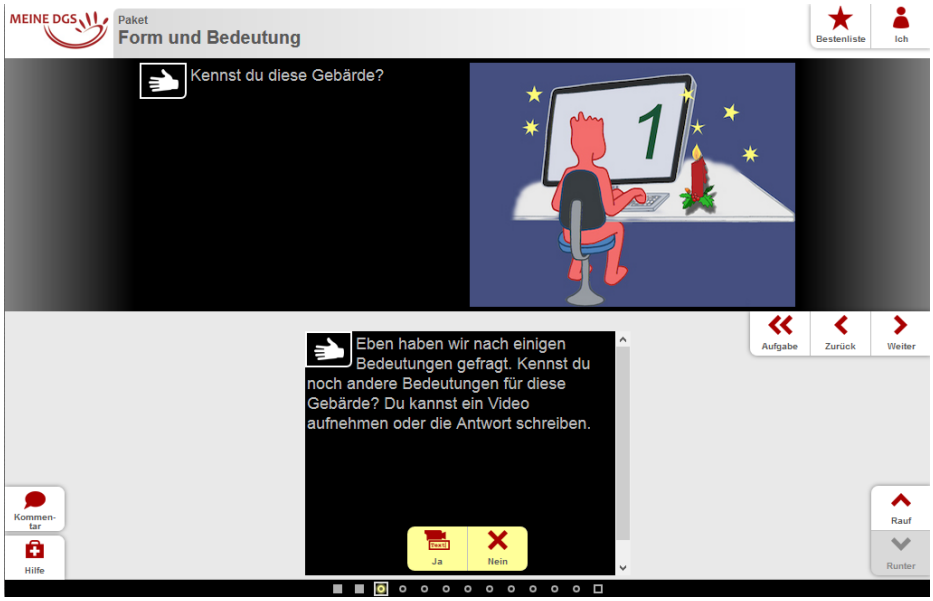


Figure 30: Screen 3 index 99 / further meanings of a sign

The special content row at the bottom is specified by the special index of 99.
By having a more detailed look at the following code we can see that there is one further special row corresponding to the construct of row index = 99.

```
<row index="99" type="buttons">
    <content type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.webm" />

        <text>
            Eben haben wir nach einigen Bedeutungen gefragt. Kennst du noch andere Bedeutungen für diese Gebärde?
            Du kannst ein Video aufnehmen oder die Antwort schreiben.
        </text>
    </content>
    <button index="0" type="option" icon="VIDEO_TEXT">Ja</button>
    <button index="1" type="option" icon="NO" goto="exit-page">Nein</button>
</row>
<row id="1183-supp" type="video-and-text">
    <content index="0" type="multimedia" hratio="360" vratio="270"/>
</row>
</page>
```

Listing 31: ROW XML and finished page tag</page>

If a user decides to give some additional information on a sign by video or text she/he is redirected to a special row with id=”1183-supp” (in this example) where 1183 corresponds to the current page id attribute. This is the area to send text answers or video answers by webcam.

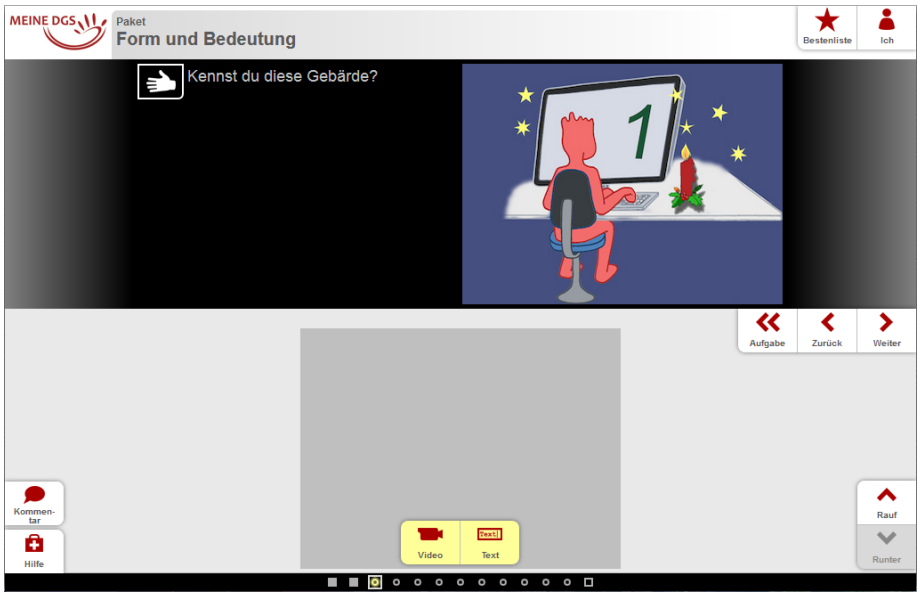


Figure 31: Area text & video | row with id 1183-supp (pageid-supp)

As you can see in the next screen the text-/video answer opens in the same iFrame of the 1183-supp row.

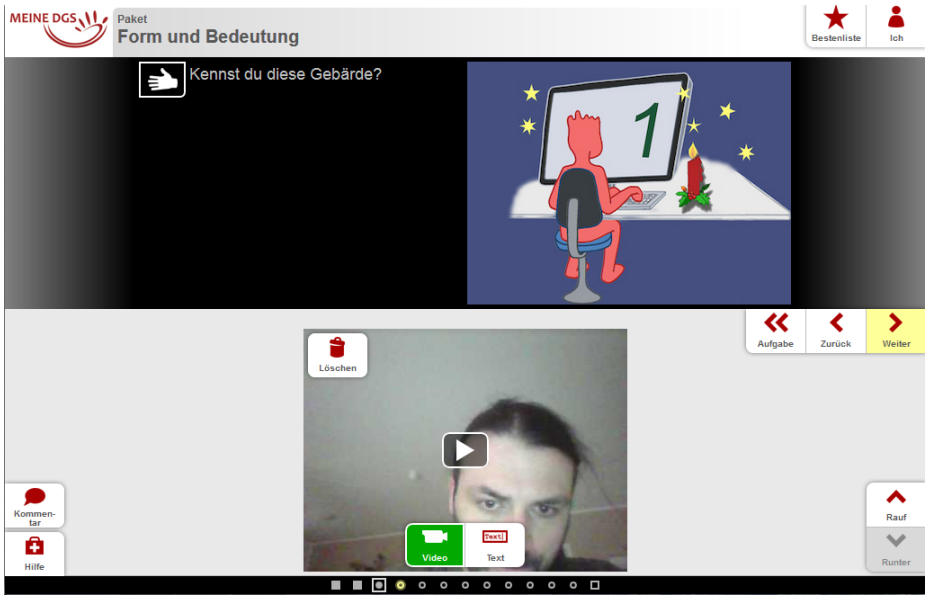


Figure 32: Video comment on further meanings of a given sign

Finally we consider an example where three components are presented side by side in one row. The three components are: Sign (left) | Text (middle) | Explanation (right)

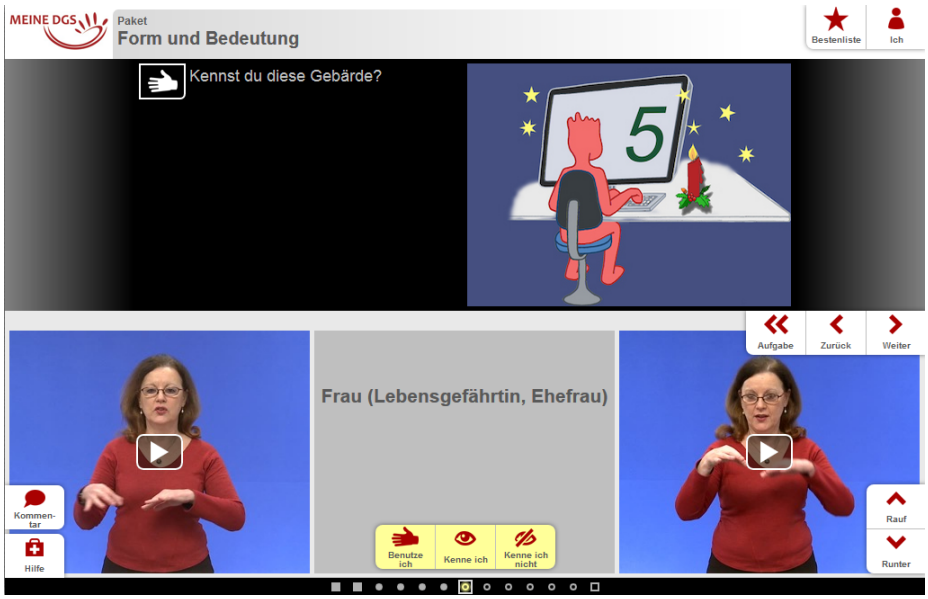


Figure 33: Three side by side components of a single row

This construct is reflected in the code as follows:

```
<row id="4032" type="buttons">

    <!-- Content on the left -->
    <content id="4032" index="7" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.webm" />

    </content>

    <!-- Content in the middle -->
    <content index="1" type="keyword" hratio="360" vratio="270">Frau (Lebensgefährtin, Ehefrau)</content>

    <!-- Content on the right -->
    <content index="2" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.webm" />

    </content>

    <button id="4032-2" index="0" type="option" icon="SIGN_USED">Benutze ich</button>
    <button id="4032-1" index="1" type="option" icon="SIGN_KNOWN">Kenne ich</button>
    <button id="4032-0" index="2" type="option" icon="SIGN_UNKNOWN">Kenne ich nicht</button>
</row>
```

Listing 32: The components side by side

As you can see in the XML-Comments inside the file above, there are three content blocks available with different IDs.

Commit-page, revise-page, retract-page and help
In addition to the <page></page> construct there are four more simple areas available in the xml packidge. Namely commit-page, revise-page, retract-page and help.
Since the structure of these areas is very simple we do not have to regard a screenshot for every example here available but only for the commit page in order to illustrate the relation to aforementioned concepts.

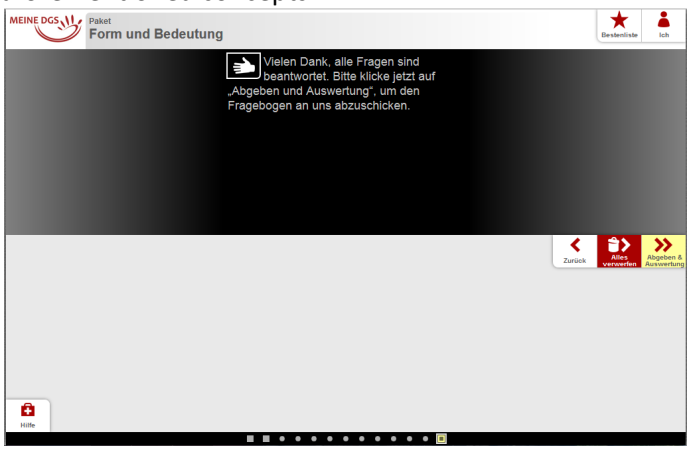


Figure 34: Commit page

The commit page enables the user to submit an answered questionnaire or to skip it. The GUI area shows a now familiar structure encoded in XML as follows:

```
<commit-page>
    <comment>
        <content index="0" type="multimedia" hratio="360" vratio="270" />
    </comment>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.webm" />

        <text>
            Vielen Dank, alle Fragen sind beantwortet. Bitte klicke jetzt auf „Abgeben und Auswertung",
            um den Fragebogen an uns abzuschicken.
        </text>
    </content>
</commit-page>
```
Listing 33: commit-page

The revise page is structured in the same way.

```
<revise-page>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.webm" />

        <text>
            Du bist am Ende des Fragebogens angekommen, aber es gibt Fragen, die du noch nicht beantwortet hast.
            Bitte gehe zurück zur ersten offenen Frage.
        </text>
    </content>
</revise-page>
```

Listing 34: revise-page

The same structure can be found with the retract page.

```
<retract-page>
    <content index="0" type="multimedia" hratio="360" vratio="270">

        <image
            alt="Standbild aus dem Gebärdenvideo"
            src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.jpg"
            srcset="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.webm" />

        <text>
            Du hast auf „Alles verwerfen" geklickt. Das bedeutet, dass alle deine Angaben gelöscht werden.
            Möchtest du das wirklich? Wenn nicht, klicke auf „Zurück".
        </text>
    </content>
</retract-page>
```

Listing 35: retract page – the same structure as the previous examples

Help area

Whereas the aforementioned structures are quiet identical the help area of the package file is structured in a different way. As you are probably aware, the help area comes in two flavours in the GUI:

Help-videos and help-texts. Compare Figures 35 and 36. The help-area starts with <help> tag inside the XML code.

Figure 35: Video based help area in the frontend GUI

Figure 36: Text-based help in the GUI

In order to become aware of how to place new help components for a packidge let us have a glimpse on the corresponding code that is:

```
    <help>
        <content-ref id="goldener pfad" />
        <content-ref id="text gebaerde" />

        <content-ref id="antwortknoepfe" />
        <content-ref id="kommentar" />

        <content-ref id="videoupload" />
        <content-ref id="kamera" />

        <content-ref id="kamera_firefox" />
        <content-ref id="kamera_chrome" />

        <content-ref id="abgeben" />
        <content-ref id="abmelden" />

    </help>
</packidge>
```

Listing 36: Help tag in a packidge file

As you can see the help contents are only references to a corresponding content id in the “help.xml” file. The real contents available for a usergroup (role) are defined in the “help.xml” file in the role directory itself.

8.3 Metadata related questionnaires (Additional features by examples / Regular Expressions I)

In contrast to content related questionnaites that reside inside the packidge-directory these are stored in the role folder so that the profiles can be defined in dependency of the roles (i.e. in different languages).

8.3.1 packidge.CHANGE_PROFILE.xml (User profile change)

Metadata related questionnaires became introduced in chapter 8. They handle profile- and registration master file data. Although the purpose is different from a content related questionnaire they have common xml construct. Technically all the given information in the previous chapter can be adapted to the metadata related questionnaires as well.
This is especially important because content related packidges use the same concepts and components too that are indroduced subsequently in this chapter. According to Listing 37 we can regard an analog structure in regard to the content related packidges.

```
<?xml version="1.0" encoding="UTF-8"?>
<packidge id="CHANGE_PROFILE" userprofile="change-profile" score="0"  >
    <topic>Benutzerprofil</topic>
    <name>Änderung</name>

    <page index="1" topic="willkommen">

    <page index="3" topic="passwort (neu)" processed="true">

    <page index="4" topic="mail" processed="true">

    <page index="5" topic="nachname" processed="true">

    <page index="7" topic="wohnort" processed="true">

    <page index="8" topic="hoerstatus" processed="true">

    <page index="9" topic="dgs" processed="true">

    <page index="10" topic="andere ga" processed="true">

    <commit-page>

    <revise-page>

    <retract-page>

    <help>

</packidge>
```

Listing 37: Same structure as always

Having a look at the GUI presenting the user profile change process we can identify a well known layout structure from the previous chapters.

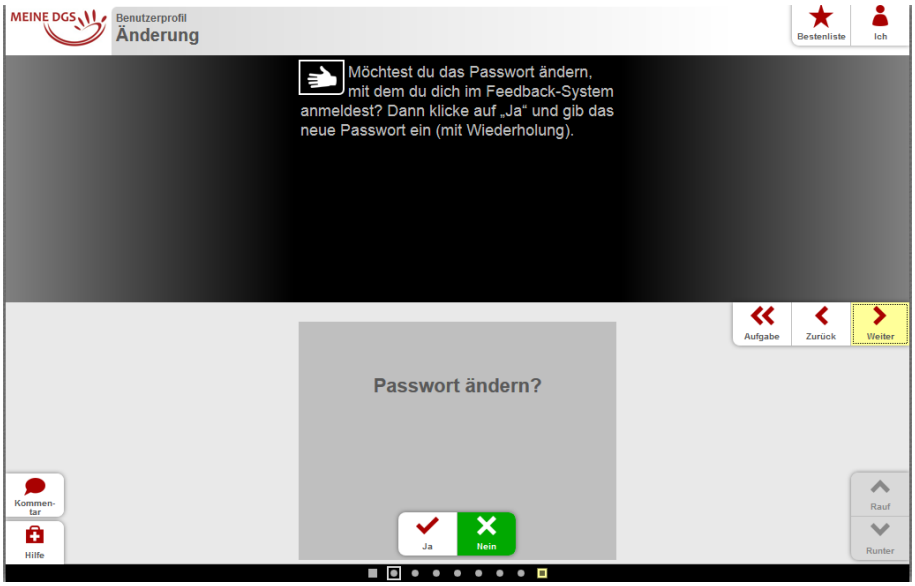


Figure 37: User profile change

Nevertheless there are some additional features that can be presented at this point that can be used in order to create new questionnaires of any type. The focus in this chapter lies on the usage and integration of regular expressions that occur in context of field validation and consistency checks. Regarding the password change process illustrated by the figure above we are faced with a basic xml structure as follows.

```
<row index="0" type="buttons">
    <content index="0" type="keyword" hratio="360" vratio="270">Passwort ändern?</content>
    <button index="0" type="option" icon="YES">Ja</button>
    <button index="1" type="option" selected="true" icon="NO" goto="exit-page">Nein</button>
</row>
```

Listing 38: The “NO” value is prechecked on initial load of the row

By changing the password the application checks wheter the entered string is valid in regard to the password security policy that is implemented by a regular expression.

Figure 38: String validation on password change 1

A regular expression for validation and consistency checks is reference by the valutype attribute in the row tag.

Figure 39: String validated successfully

As of listing 39 the valuetype has been correctly set to “!PASSWORD” in this case. Please have a look at chapter 11 for more details on the application’s regular expression definitions for text field validation.

```
<row index="1" id="password" type="value" valuetype="!PASSWORD" autocomplete="new-password">
    <label>Neues Passwort</label>
</row>
```

Listing 39: REGEX – valuetype password

These definitions also define an EMAIL valuetype that is represented by the next example.

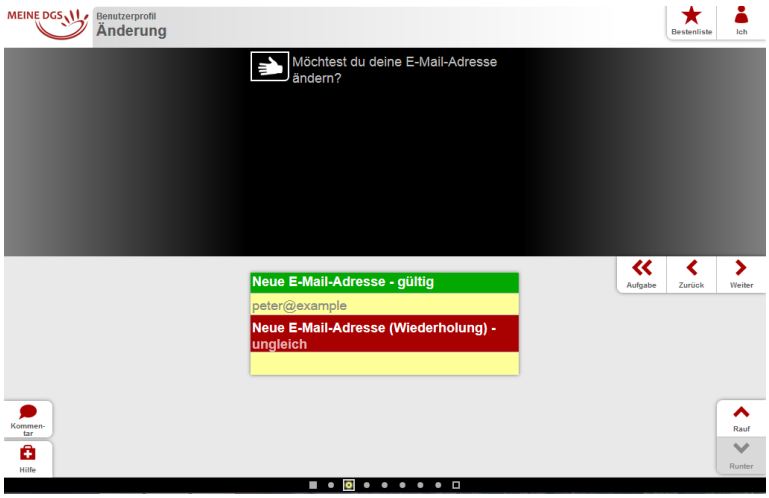


Figure 40: Regular expression – Only @-sign is checked for a valid email address

In this case the usage of an @-character is mandatory (valuetype “!EMAIL”).

```
<row index="3" type="value" valuetype="!EMAIL" autocomplete="email">
    <label>Neue E-Mail-Adresse</label>
</row>
```

Listing 40: Corresponding xml / REGEX is evaluated according to the valuetype attribute

The following construct represents a REGEX called NUMBER as it is decribed in chapter 11. In addition a maximum quantity of five digits are defined. How this works is explained in chapter 8.4.1.

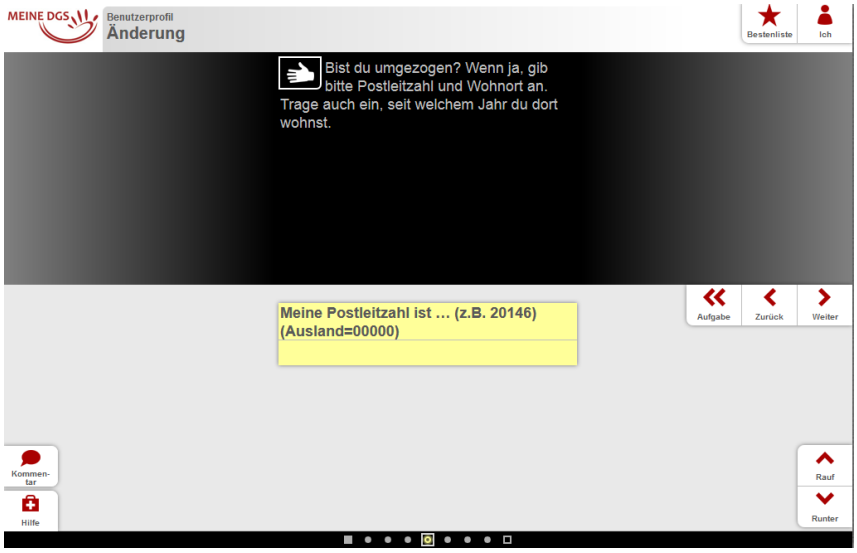


Figure 41: Regular Expression – Integer Value – 5 digit tokens

A further available regular expression definition commonly used in the application is YEAR as it is shown by figure 42. Compare chapters 8.4.1 and 11.

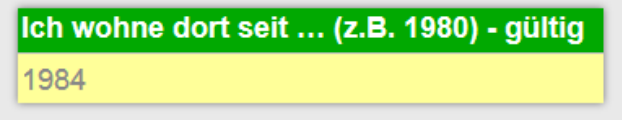


Figure 42: REGEX year

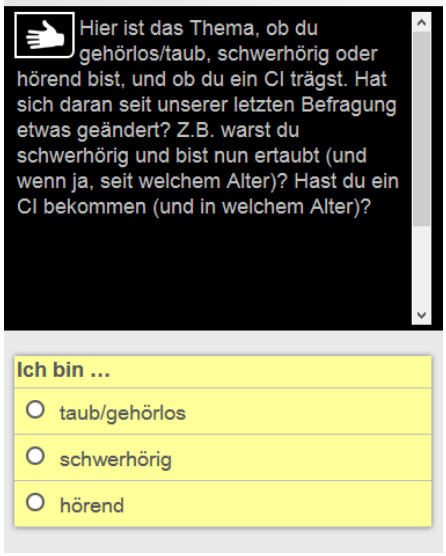


Figure 43: Selection list

We will have a more detailed look on regular expressions in context of the PROFILE.xml package in chapter 8.3.2. Before that, some further GUI elements are introduced which can be used in self-created questionnaires as well.
Taking a look at figure 43 we can see a selection list as it is presented in the web GUI.

If you like to use your individual list you have to be geared to the structure of the following listing.

```
<row index="1" type="options">
    <label>Ich bin …</label>
    <option type="exclusive">taub/gehörlos</option>
    <option type="exclusive" goto="3">schwerhörig</option>
    <option type="exclusive" goto="4">hörend</option>
</row>
```

Listing 41: Selection type exclusive – only one item selectable

In order to make it possible to let users select only one item per list the options type attribute has to be set to the value “exclusive”. Another important attribute in this context is the goto-attribute which takes a certain integer index as parameter value. This attribute is used in order to direct users to a special area of the questionnaire depending on a pre-condition. This could be in dependency on a certain selected item of a list. In the following listing it only makes sense for a user to answer a question about her/his amblyacousia if the value “deaf” was selected in a preceding step for example.

```
<row index="0" type="buttons">
    <content index="0" type="keyword" hratio="360" vratio="270">Angaben zum Hören ändern?</content>
    <button index="0" type="option" icon="YES">Ja</button>
    <button index="1" type="option" selected="true" icon="NO" goto="exit-page">Nein</button>
</row>
<row index="1" type="options">
    <label>Ich bin …</label>
    <option type="exclusive">taub/gehörlos</option>
    <option type="exclusive" goto="3">schwerhörig</option>
    <option type="exclusive" goto="4">hörend</option>
</row>
<row index="2" type="value" valuetype="AGE" goto="4">
    <label>Taub/gehörlos seit ich … Jahre alt bin.</label>
</row>
<row index="3" type="value" valuetype="AGE">
    <label>Schwerhörig seit ich … Jahre alt bin.</label>
</row>
<row index="4" type="buttons">
    <content index="0" type="keyword" hratio="360" vratio="270">Ich trage ein CI:</content>
    <button index="0" type="option" icon="YES">Ja</button>
    <button index="1" type="option" icon="NO" goto="exit-page">Nein</button>
</row>
```

Listing 42: Row index is incremented / use of the goto attribute as link

The goto attribute references rows that are presented depending on the preceding selections. In the case of goto=”3” in this example the user is aked to enter her/his age.

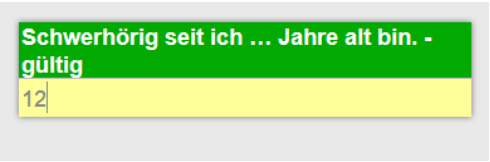


Figure 44: Example of goto=”3” / REGEX AGE

As mentioned before this is when regular expressions come into play. In case of figure 44 an AGE value is checked for consistency. Please note that the valuetype in listing 42 is of type AGE. Compare chapter 11.
In addition to an exlusive selection list that has been discussed in this chapter there are multiple selection list available for Feedback questionnaire development. One example is presented in figure 45 where two items have been checked at the same time.

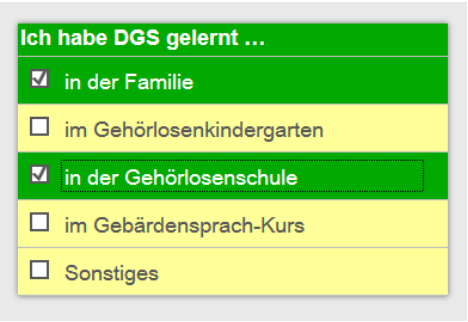


Figure 45: Example of a multiple selection

This behaviour is implemented simply through omitting the type parameter in an options tag.

```
<row index="1" type="options">
    <label>Ich habe DGS gelernt …</label>
    <option goto="3">in der Familie</option>
    <option goto="3">im Gehörlosenkindergarten</option>
    <option goto="3">in der Gehörlosenschule</option>
    <option goto="3">im Gebärdensprach-Kurs</option>
    <option>Sonstiges</option>
</row>
```

Listing 43: option tag for multiple selection

It is also possible to combine both types of lists.

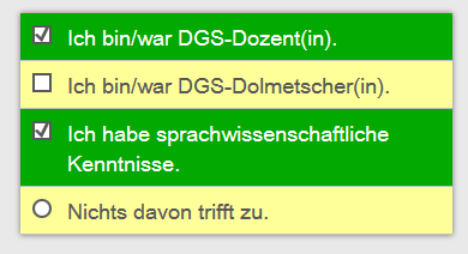


Figure 46: Combined selection elements / exclusive type and non-exclusive type

By adding the type=”exclusive” parameter to the last option an exclusively selectable item becomes integrated in a multiple selection list.

```
<row index="4" type="options">
    <option>Ich bin/war DGS-Dozent(in).</option>
    <option>Ich bin/war DGS-Dolmetscher(in).</option>
    <option>Ich habe sprachwissenschaftliche Kenntnisse.</option>
    <option type="exclusive">Nichts davon trifft zu.</option>
</row>
```

Listing 44: Both list types combined

8.3.2 packidge. PROFILE.xml (User profile – Personal data)

In the PROFILE questionnaire the regular expression construct “NAME” is applied in the following way.

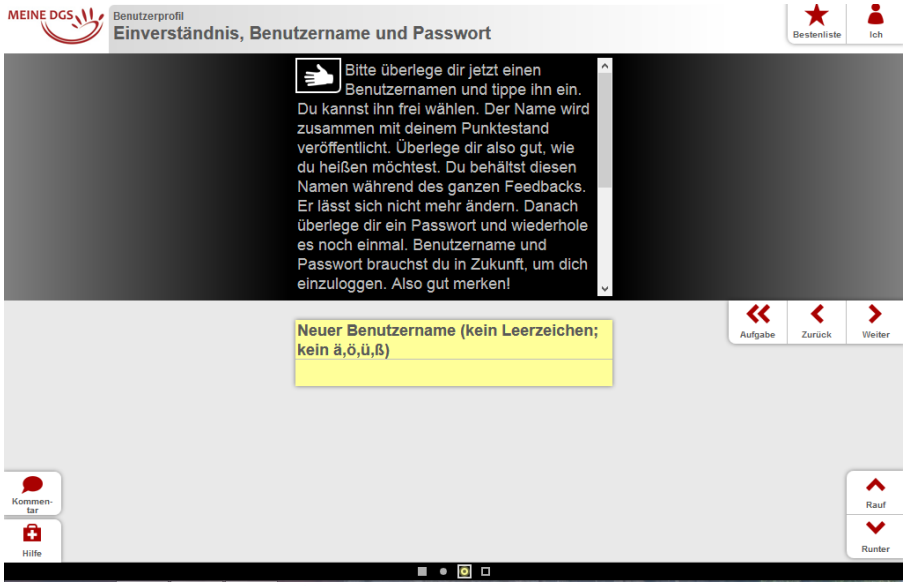


Figure 47: REGEX – new user- / nickname

Thereby the NAME-REGEX is used for different types of text fields such as usernames or lastname fields as follows.

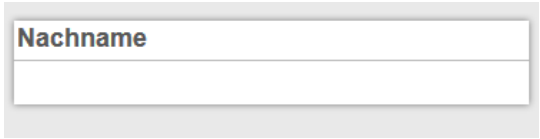


Figure 48: Last name REGEX: valuetype: NAME

The implementation of the text fields contains the appropriate valuetype.

```
<row index="2" type="value" valuetype="NAME">
    <label>Nachname</label>
</row>
```

Listing 45: REGEX valuetype NAME

Every regular expression can be used in different contexts. The final example shows the REGEX YEAR this time used for “year of birth”.

```
<row index="1" type="value" valuetype="YEAR:-120,-18">
    <label>Mein Geburtsjahr ist … (z.B. 1975)</label>
</row>
```

Listing 46: valuetype YEAR

8.3.3 packidge.REGISTRATION_PROFILE.xml (User profile – Registration)

No further xml constructs are occuring in this questionnaire type.

8.4 Additional Features / Regular Expressions II

In order to resume how to apply regular expressions on Feedback questionnaires the INPUT-packidge is presented. This package originates from the software testing process of Feedback. By means of this package the entire power of regular expressions in context of the application becomes clear.

8.4.1 INPUT-packidge

The valuetype attribute contains versatile occurrences of regular expressions that can be useful to verify entries from the GUI. Among many examples in the following listing there are text specific expressions as follows:

```
<page index="1">
  <comment />
  <content type="keyword">Werteingabe / Validierung</content>
  <row id='freetext' type='value' valuetype='FREE_TEXT'><label>Freitext (mehrzeilig)</label></row>
  <row id='text' type='value' valuetype='TEXT'><label>Text (einzeilig)</label></row>
  <row id="text-min10" type="value" valuetype="TEXT:10,"><label>Text (min. Länge: 10)</label></row>
  <row id='text-max3' type='value' valuetype='TEXT:,3'><label>Text (max. Länge: 3)</label></row>
  <row id='text-3' type='value' valuetype='TEXT:3,3'><label>Text (exakte Länge: 3)</label></row>
  <row id='typotext' type='value' valuetype='!TEXT'><label>Beliebiger Text, doppelt</label></row>
  <row id='name-max20' type='value' valuetype='NAME:,20'><label>Name (max. Länge: 20)</label></row>
  <row id='number-42-4711' type='value' valuetype='NUMBER:42,4711'><label>Nummer (min. 42, max. 4711)</label></row>
  <row id='number-min1000' type='value' valuetype='NUMBER:1000,'><label>Nummer (min. 1000)</label></row>
  <row id='year' type='value' valuetype='YEAR'><label>Jahr</label></row>
  <row id='year-back10' type='value' valuetype='YEAR:-10,0'><label>Jahr (min. vor 10 Jahren, max. heute)</label></row>
  <row id='username' type='value' valuetype='USER_NAME'><label>Benutzername</label></row>
  <row id='password' type='value' valuetype='!PASSWORD'><label>Passwort</label></row>
  <row id='email' type='value' valuetype='!EMAIL'><label>Email</label></row>
  <row id='regex-abc' type='value' valuetype='REGEX:[ABC]*'><label>Regex (nur As, Bs und Cs)</label></row>
  <row id='regex-plz' type='value' valuetype='REGEX:(D-)?[0-9]{5}'><label>Regex (deutsche PLZ)</label></row>
</page>
```

Listing 47: Implementation of versatile valuetypes as REGEX XML attributes

TEXT → text (single line)
TEXT:10 → minimum length 10
!TEXT → random text duplex
TEXT:3,3 → difinite length 3
NAME:,20 → name: maximum length 20
Further defenitions can be found in chapter 11.

8.4.2 Validation of lists, scale questions etc.

Besides the commonly used button based navigation there are further row navigation elements that can be used in order to answer questions. If a question is not supossed to be anwered by a simple "yes" or "no", questionnaire developers have the opportunity to use scale questions.
The application provides special GUI elements for this purpose that can be seen in the next examples.

Figure 49: Example of scale questions

In order to create the scale in figure 49 a developer has to implement a questionnaire scale element by defining row of type="scale" according to listing 48.

```
<row index='1' type='scale'>
  <content index='1' type='keyword' hratio='4' vratio='3'>Ich gebärde und verstehe DGS...</content>
  <button index='0' type='scale' icon='THUMB' iconindex='6'>wie Muttersprachler</button>
  <button index='1' type='scale' icon='THUMB' iconindex='5' />
  <button index='3' type='scale' icon='THUMB' iconindex='3' />
  <button index='4' type='scale' icon='THUMB' iconindex='2' />
  <button index='6' type='scale' icon='THUMB' iconindex='0'>wenig/keine Kenntnisse</button>
</row>
```

Listing 48: Implementation of row type scale

A developer is free in her/his choice which icons to use for the scale. In the Feedback context it is also possible to use smiley icons as follows.

Figure 50: Another smiley scale / same implementation paradigm

Depending on the row type different layouts and semantical constructs can be implemented into a packidge. Further row types are:

- buttons

```
<row index="3" type="buttons">
    <content index="0" type="multimedia" hratio="640" vratio="480">
        <image src="https://feedback.sign-lang.uni-hamburg.de/xxx.png"/>
        <video src="https://feedback.sign-lang.uni-hamburg.de/xxx.mp4"/>
        <video src="https://feedback.sign-lang.uni-hamburg.de/xxx.webm"/>
        <text>*** Trägst du ein CI? ***</text>
    </content>
    <button index="0" type="option" icon="YES">Ja</button>
    <button index="1" type="option" icon="NO">Nein</button>
</row>
```

Listing 49: Buttons type

- value

```
<row index='2' type='value' valuetype='NAME'>
 <label>Schwerhörig seit … (z.B. 1975):</label>
 </row>
```

Listing 50: row type value

- video-and-text

```
<row index='0' type='video-and-text'>
    <content index='0' type='multimedia' hratio='1143' vratio='859'>
    </content>
</row>
```

Listing 51: Video and text type

- options

```
<row index='4' type='options'>
    <content index="0" type="multimedia" hratio="640" vratio="480">
        <image src="https://feedback.sign-lang.uni-hamburg.de/xxx.png"/>
        <video src="https://feedback.sign-lang.uni-hamburg.de/xxx.mp4"/>
        <video src="https://feedback.sign-lang.uni-hamburg.de/xxx.webm"/>
        <text>Hast du früher längere Zeit in anderen Bundesländern gelebt?</text>
    </content>
    <option type="exclusive">Nein</option>
    <option>Baden-Württemberg</option>
    <option>Bayern</option>
    <option>Berlin</option>
    <option>Brandenburg</option>
    <option>Bremen</option>
    <option>Hamburg</option>
    <option>Hessen</option>
    <option>Mecklenburg-Vorpommern</option>
    <option>Niedersachsen</option>
    <option>Nordrhein-Westfalen</option>
    <option>Rheinland-Pfalz</option>
    <option>Saarland</option>
    <option>Sachsen</option>
    <option>Sachsen-Anhalt</option>
    <option>Schleswig-Holstein</option>
    <option>Thüringen</option>
</row>
```

Listing 52: Options type

9. Answered questionnaires as results

When a questionnaire has been answered completely the global page attribute „status“ is set to the “committed” value. This is when the resulting file becomes available for evaluation in iLex. According to the example of packidge75.xml in chapter 8.2 the following results become available.

```
<?xml version="1.0" encoding="UTF-8"?>
<packidge id="75" date="2014.12.17 14:09:37" status="committed" score="24">
  <topic>Paket</topic>
  <name>Form und Bedeutung</name>
  <help>
  <page id="page0" processed="true">
  <page id="page1" processed="true">
  <page id="1183" processed="true">
  <page id="1176" processed="true">
  <page id="1174" processed="true">
  <page id="1175" processed="true">
  <page id="1178" processed="true">
  <page id="1177" processed="true">
  <page id="1182" processed="true">
  <page id="1189" processed="true">
  <page id="1191" processed="true">
  <page id="1190" processed="true">
  <revise-page id="revisePageId">
  <commit-page id="commitPageId" processed="true">
  <retract-page id="retractPageId">
</packidge>
```

Listing 53: processed attribute

As listing 53 shows each page inside a packidge has a “processed” attribute. That is why a user is able to stop and to resume the answering process of a questionnaire. On commitment of the packidge a timestamp is set and the status becomes “committed”.
Having a deeper look at a single processed page (processed=“true“) we become aware that every page of the result document also consists of contents, comments and rows as is is the case with the initial packidges that have been considered in the previous chapter.
Comments are now (after processing) referenced by unique IDs in order to link to ressources in the file system (i.e. jpg, webm). That is because the comment files itself are stored seperately inside the user-directory.

```
  <page id="1178" processed="true">
    <content id="content0" hratio="360" vratio="270" type="multimedia">
    <content id="content1" hratio="360" vratio="270" type="multimedia">
    <comment id="video-and-text-sub-page1418814467200" type="video-and-text" goto="next-sub-page">
    <row id="4027" type="buttons" goto="next-sub-page">
    <row id="4029" type="buttons" goto="next-sub-page">
    <row id="4030" type="buttons" goto="next-sub-page">
    <row id="4031" type="buttons" goto="next-sub-page">
    <row id="4033" type="buttons" goto="next-sub-page">
    <row id="4028" type="buttons" goto="next-sub-page">
    <row id="4032" type="buttons" goto="next-sub-page">
    <row id="4034" type="buttons" goto="next-sub-page">
    <row id="4035" type="buttons" goto="next-sub-page">
    <row id="buttons-sub-page1418814467200" type="buttons" goto="next-sub-page">
    <row id="1178-supp" type="video-and-text" goto="next-sub-page">
  </page>
```

Listing 54: A single, processed page

The comment in the listing above is page-global whereas the next example illustrates the deposition of comment data in the filesystem of a user by a so called “ID-suppl row comment”. This is the kind of comment that can be made in order to add some additional information on the usage of a special sign.
According to chapter 8.2 the concrete example of row id=”1183-supp” here corresponds to the <page> id attribute 1183 of the current page.

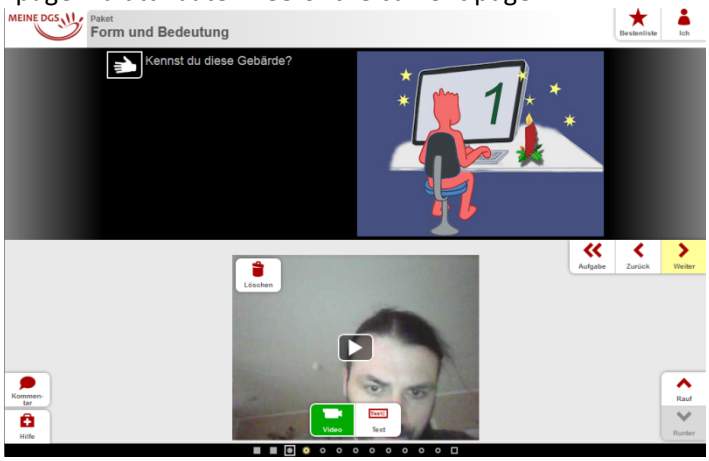


Figure 51: Making a video answer inside the –supp id area of the corresponding xml

The files of the video answers are stored in the user’s individual directory, too - together with unique IDs in their filenames. If the user’s answer is just text-based the text becomes stored inside the resulting packidge.xml file itself. That is why the package is called a package. It is all packed together in one file for further operating in iLex.
The next listing gives an insight into how the video comments are stored into the file system and get referenced by the result packidge.xml (1183-supp).

```
<row id="1183-supp" type="video-and-text" goto="next-sub-page">
  <content id="0" hratio="4" vratio="3" type="multimedia">
    <image src="sven.berding.standard-75-2-4-1418816764005.jpg" />
    <video src="sven.berding.standard-75-2-4-1418816764005.webm" />
  </content>
</row>
```

Listing 55: The src attributes are referencing file system resources

The referenced files are stored in the user’s directory as follows:

packidge.74	16.12.2014 18:36	XML-Datei
packidge.75	17.12.2014 14:09	XML-Datei
packidge.80	16.12.2014 18:49	XML-Datei
packidge.86	16.12.2014 10:48	XML-Datei
packidge.CHANGE_PROFILE_14A52810EDC	16.12.2014 10:52	XML-Datei
packidge.CHANGE_PROFILE_14A58658E70	17.12.2014 14:17	XML-Datei
packidge.REGISTRATION_PROFILE	16.12.2014 10:32	XML-Datei
sven.berding.standard-74-4-4-1418750879099	16.12.2014 18:31	IrfanView JPG File
sven.berding.standard-74-4-4-1418750879099	16.12.2014 18:31	VLC media file (.w...
sven.berding.standard-75-2-4-1418816764005	17.12.2014 13:35	IrfanView JPG File
sven.berding.standard-75-2-4-1418816764005	17.12.2014 13:35	VLC media file (.w...
user-config	16.12.2014 10:52	XML-Datei
user-status	17.12.2014 14:09	XML-Datei
user-trace	19.12.2014 11:56	XML-Datei

Figure 52: Resourses in the file system

Among these files the referenced data can be found.
sven.berding.standard-75-2-4-1418816764005.jpg (Preview image)
sven.berding.standard-75-2-4-1418816764005.webm (Video comment)
Comments are available for pages on a global level or as we can see here as further information on a sign. The third occurance of a comment can be found on the commit page.
Having a detailed look on a simple row we become aware of how the resulting packidge file marks the certain user's button clicks. Depending on the button selected by the user the xml code contains a boolean marker. If a button was selected (i.e. the YES button in the following example) an attribute „selected" becomes added to the button tag and is set to true.
If a button is not clicked at all this attribute is missing.

```
<row id="4027" type="buttons" goto="next-sub-page">
  <content id="4027" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4027/2300908.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4027/2300908.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4027/2300908.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4027/2300908.webm" />
  </content>
  <button icon="YES" selected="true" goto="next-sub-page">Ja</button>
  <button icon="NO" goto="exit-page">Nein</button>
</row>
```

Listing 56: Buttons

Considering the evaluation of a question dealing with the knowledge of a sign, we can find the identical way of processing boolean expressions (cf. listing 57). The button with the icon SIGN_USED holds an attribute „selected" that was set to true as follows:

```
<row id="4029" type="buttons" goto="next-sub-page">
  <content id="4029" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4029/2300920.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4029/2300920.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4029/2300920.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4029/2300920.webm" />
  </content>
  <content id="content1" hratio="360" vratio="270" type="keyword">heiraten</content>
  <button icon="SIGN_USED" selected="true" goto="next-sub-page">Benutze ich</button>
  <button icon="SIGN_KNOWN" goto="next-sub-page">Kenne ich</button>
  <button icon="SIGN_UNKNOWN" goto="next-sub-page">Kenne ich nicht</button>
</row>
```

Listing 57: Boolean was set to true here as well → selected="true"

Even if there are further content blocks in a row the button handling remains the same as you can see in the following code snippet.

```
<row id="4032" type="buttons" goto="next-sub-page">
  <content id="4032" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2300918.webm" />
  </content>
  <content id="content1" hratio="360" vratio="270" type="keyword">Frau (Lebensgefährtin, Ehefrau)</content>
  <content id="content2" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1178/4032/2308849.webm" />
  </content>
  <button icon="SIGN_USED" selected="true" goto="next-sub-page">Benutze ich</button>
  <button icon="SIGN_KNOWN" goto="next-sub-page">Kenne ich</button>
  <button icon="SIGN_UNKNOWN" goto="next-sub-page">Kenne ich nicht</button>
</row>
```

Listing 58: Identical button evaluation

The revise page remains untouched in the result document. There is no user information added here.

```
<revise-page id="revisePageId">
  <content id="content0" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/revise.webm" />
    <text>
      Du bist am Ende des Fragebogens angekommen, aber es gibt Fragen, die du noch nicht beantwortet hast.
      Bitte gehe zurück zur ersten offenen Frage.
    </text>
  </content>
</revise-page>
```

Listing 59: revise page stays the same

In order to present the status of the commit page a “processed” attribute becomes available here too. Please note the occurrence of the comment section at this point (Listing 60).

```
<commit-page id="commitPageId" processed="true">
  <content id="content0" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/danke.webm" />
    <text>Vielen Dank, alle Fragen sind beantwortet. Bitte klicke jetzt auf „Abgeben und Auswertung“,
  </content>
  <comment id="video-and-text-sub-page1418814467200" type="video-and-text" goto="next-sub-page">
    <content id="0" hratio="4" vratio="3" type="multimedia" />
  </comment>
</commit-page>
```

Listing 60: Commit Page, Processed=“true“

The retract page stays untouched after processing as well:

```
<retract-page id="retractPageId">
  <content id="content0" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/allgemein/retract.webm" />
    <text>Du hast auf „Alles verwerfen“ geklickt. Das bedeutet, dass alle deine Angaben gelöscht
  </content>
</retract-page>
```

Listing 61: retract page

10. Help pages

In this documentation the help page topic has already been touched here and there. In this chapter we have a quick look at the details of how to build your own help files. As mentioned before the "help.xml" file is the central file for creating own help contents. These contents can be created inside the role directory in order to create help contents for a specific group of users. These help contents can be referenced from the packidges selectively.
The help.xml is structured as follows:

```
<help>
	<content id='overview' type="multimedia" hratio="640" vratio="480">
		<image src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_1.png"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_1.mp4"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_1.webm"/>
		<text>This is an overview help text!</text>
	</content>
	<content id='buttons' type="multimedia" hratio="640" vratio="480">
		<image src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_2.png"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_2.mp4"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_2.webm"/>
		<text>This is a help text on button usage!</text>
	</content>
	<content id='video' type="multimedia" hratio="640" vratio="480">
		<image src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_3.png"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_3.mp4"/>
		<video src="http://feedback.sign-lang.uni-hamburg.de/Typ1/hilfe_3.webm"/>
		<text> is a help text on video recording!</text>
	</content>
</help>
```

Listing 62: help.xml

The help contents are referenced in the packidge.xml in the following manor:

```
<help>
<content-ref id='video' />
<content-ref id='buttons' />
</help>
```

Listing 63: help references in packidge.xml

As we can see in the previous example the overview section of the "help.xml" is not referenced in the listing 63. So help contents can be integrated selictively in the packidges. The order of the items in the packidge web GUI corresponds to the order of items in the help section of a packidge. The global application help relates to the order in the help.xml file.
Finally we will take a look at a concrete example of packidge76.xml

```
<help>
<content-ref id="goldener pfad" />
<content-ref id="text gebaerde" />

<content-ref id="antwortknoepfe" />
<content-ref id="kommentar" />

<content-ref id="videoupload" />
<content-ref id="kamera" />

<content-ref id="kamera_firefox" />
<content-ref id="kamera_chrome" />

<content-ref id="abgeben" />
<content-ref id="abmelden" />

</help>
```

Listing 64: help contents of packidge76

This is how the help contents are integrated for productional use.

11. Validation of text fields

User entries are validated according to the datatype mentioned in the XML valuetype.
A datatype is defined in the XML in the following form:
1. Name
2. optional: One or many datatype-specific parameters in parenthesis.
Examples:
TEXT, NUMBER, NUMBER(-100, 100), REGEX([A-Z]*), AGE
Put an exclamation mark in front of the validation type in order to query a value twice, i.e. "!USER_NAME".
For multiline texfields compare FREE_TEXT.

Datatypes:
TEXT:
Meaning: random single-line text.
Valid tokens: No restrictions.
Regular Expression: .*
Parameters:
0 Parameters: Any length
Example: "TEXT"
2 numeric parameters: min length + max length
Examples:
"TEXT(,20)" --> max. 20 tokens
"TEXT(5,)" --> min. 5 tokens
"TEXT(1,5)" --> between 1 bis 5 tokens.

FREE_TEXT: Like TEXT but multiline.

NAME:
Meaning: one (Plasce-, Person-, or other) name.
Valid tokens: Letters (incl. umlauts), digits, point and hyphen.
Regular Expression: [a-zA-ZÀ-ÖØ-öø-ž][0-9a-zA-ZÀ-ÖØ-öø-ž\.\- ']*
Parameter: Like TEXT.

USER_NAME:
Meaning: User name for the Meine-DGS-System.
Valid tokens: Letters, digits, minus, underscore, point, space,
apostrophy, special letters of other languages/special characters.
1. sign has to be a letter or a digit.
Min-Length: Three tokens.
Speciality: Will be compared to a server-side user name.
Regular Expression: [a-zA-Z0-9][a-zA-Z0-9-_\\.]{2,}
Parameters: None.

PASSWORD:
Meaning: Password of a user of the Meine-DGS-Systems.
Valid tokens: No restrictions.
Mindestlänge: Six tokens.
Regular Expression: .*

Parameters: None.

EMAIL:
Meaning: A random Email-Adress.
Valid tokens: has to contain a @-sign, no further restrictions.
Min-Length: Three tokens.
Regular Expression: .+@.+
Parameters: None.

REGEX:
Meaning: A text, whose syntactical structure is determined by a regular expression.
Valid tokens: Depending on the regular expressioin that has been added as parameter.
Parameters:
1 regular expression: Determines the valid tokens / syntax.
Examples:
"REGEX([0-9]{5}" --> five-digit PLZ/ZIP
"REGEX([A-Z]*)" --> Upper case letters without german umlauts.
"REGEX(.*)" --> Like TEXT
"REGEX(.*{3,10})" --> Like TEXT(3,10), min. 3, max. 10 tokens

NUMBER:
Meaning: A random integer value.
Valid tokens: Digits, when indicated leading minus sign followed my at least 1 digit.
Threshold value: Depending on the present parameters.
Parameters:
0 Parameter: random integer number.
Example: "NUMBER"
2 numeric parameters: Number between min and max value (incl. both threshold values)
Examples:
"NUMBER(-100, 100)" --> Number from -100 to 100
"NUMBER(,4711)" --> Number up to max. 4711
"NUMBER(100,) --> Number from min. 100

YEAR:
Meaning: A random year date.
Valid tokens: Like NUMBER.
Threshold values: Dependend on the present parameters.
Parameters:
0 Parameter: random year.
2 numeric parameters: Begin and end of the time interval in years related to the current year.
Examples:
"YEAR(-100, 1)" --> 100 years ago until the next year (relative to the current year)
"YEAR(1914,)" --> in 1914 years(!) util infinity
"YEAR(-100,0) --> 1914 (thus: 100 years ago) up to infinity
"YEAR(0,20)" --> today up to 20 years

AGE:
Meaning: a statement of age.
Valid tokens: Digits.
Threshold values: From 0 to 120 years.
Parameters: None.

12. Loose coupling between Feedback and iLex

In order to access the Feedback components of iLex it is nescessary to activate the Feedback module inside of iLex first. Therefore the checkbox "Use Feedback Module" has to be checked under the following path:
iLex --> Preferences --> Feedback --> Use Feedback Module
The components that are in focus of the following chapters are located under the “Data” menu item. These are

- Feedback Packages
- Feedback Requests
- Under “Parameters” the items starting with the "Feedback" prefix. These are the point of access to the user management, system packages, movie assets etc.

Please note: If some changes have been made to Feedback artefacts or if they are newly created inside of iLex they are not available for Feedback until `Data --> Deploy Feedback` has been executed or an automatic deployment has been configured.
Disambiguation
In context of the terms "pages" and "rows" that have been introduced throughout this book the terms "request" and "request item" are used in the following sections in order to describe the identical concept from an iLex point of view.
In the following context the term "request level" means the level of a Feedback page whereas "request item level" refers to the Feedback row concept.
As we are aware the feedback system is a server-side web app running under Tomcat that autonomously stores its data (in XML format) in the server’s file system. Integration with the database-centered iLex is achieved by running an iLex instance that has access to the Tomcat server’s file system, regularly running a batch that both collects data from the feedback system and delivers new data to be used by the feedback system while the feedback system is running.
The feedback system delivers packages (called “packidges“) to the user (cf. chapter 2 - Disambiguation). Each package contains several pages, each of which typically consists of several “rows”. A row is to be answered by the user, e.g. by clicking a button. Typically, lots of packages are created following the same template (questionnaire type). Packages answered by the user are stored in a user-specific directory on the server (in an XML file merging the answers into the package data), along with movies the user has produced to provide additional data.

12.1 Feedback Database Tables

In iLex, five tables contain all the information to drive the feedback system as well as to register the results:

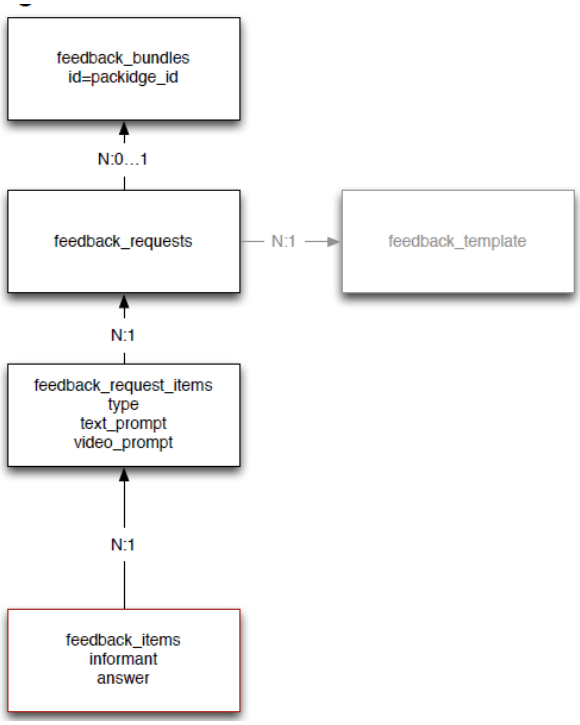


Figure 53: Entity relationships

feedback_bundles items correspond to packidges in the feedback system, whereas *feedback_requests* roughly correspond to pages and *feedback_request_items* to rows.
On the request side, the *feedback_requests* table is most central, whereas on the results side there is only the level of individual items, *feedback_items*.
Considering the following table structure from the iLex database we can see the corresponding part to the Feedback XML packidges on the web-application side. Some familiar parts of the packidges can be found inside the database table structure here such as the “name” column referring to the <name> node inside the Feedback xml. As we will see later on this entry becomes also visible in the iLex GUI layout in the context of the Feedback-Package-View.

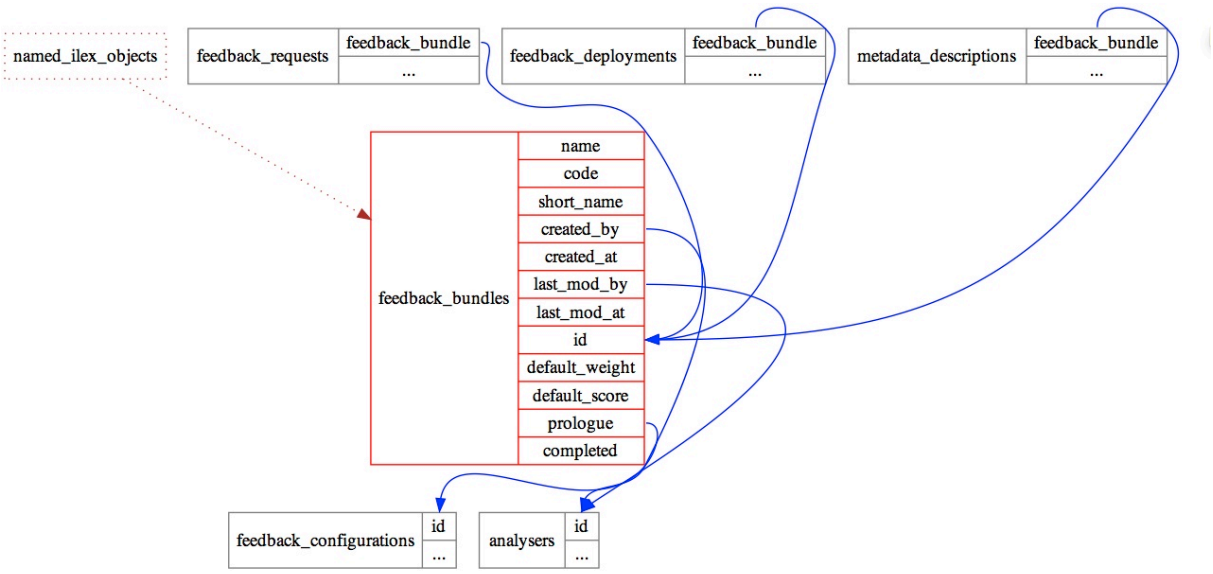
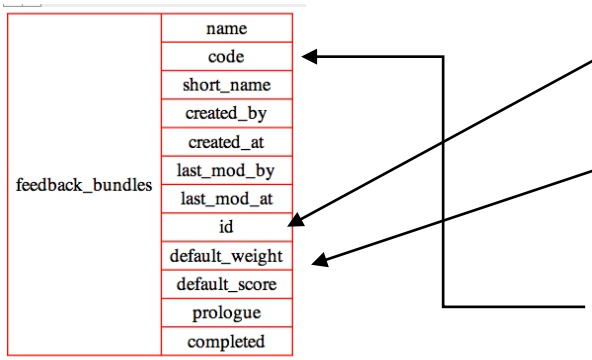


Figure 54: Feedback_bundles database context

As we can see in the database context figure above a single feedback bundle record is referenced by its id. The corresponding bundle entries in other tables are able to point to a *feedback_bundles* record easily in that way. This integrates *feedback_requests* records (which corresponds to pages in the XML package) with the *feedback_deployments* and *metadata_descriptions*.



The *id* in the left table overview is equal to the id of the <packidge> tag inside the packidge.xml.

The purpose of the *default_weight* and default_*score* are also familiar to the reader being decribed in chapter 5 “The application concept”.

The code column contains the iLex code (i.e. February month). See item “Feedback Pakages” in the iLex GUI for more details.

Figure 55: Feedback_bundles overview

As described previously a feedback_requests record references the corresponding feedback_bundles (column feedback_bundle in the figure below). As we have seen in fig. 53 before there is a N:0…1 relationship between feedback_requests and feedback_bundles (pages to packidges). In other words one xml packidge node can contain multiple page nodes which is reflected in Feedback’s web GUI since a packidge can be skimmed through.

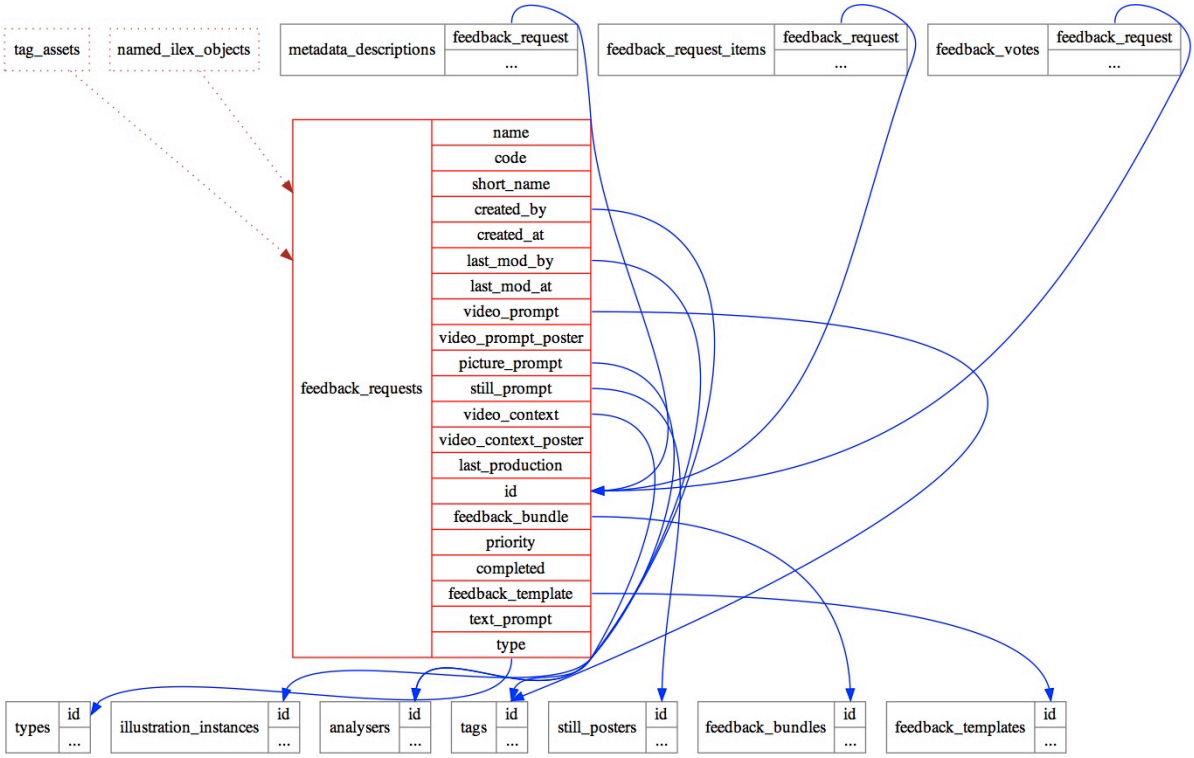


Figure 56: feedback_requests database table

As the id column in the fig. 56 above makes clear each *feedback_request* record has its own identifiier. This is namely the page id from the specific page node inside the packidge xml (i.e. <page id="1183" …>) .

In the same way a bundle can contain multiple pages one *feedback_request* can contain multiple *feedback_request_items* which are corresponding to rows inside the packidge xml.
As we can see in the database model below the *feedback_request_items* residing inside the *feedback_items* and the *feedback_votes* table are referencing their corresponding record inside the *feedback_request_items* table by a specific id.

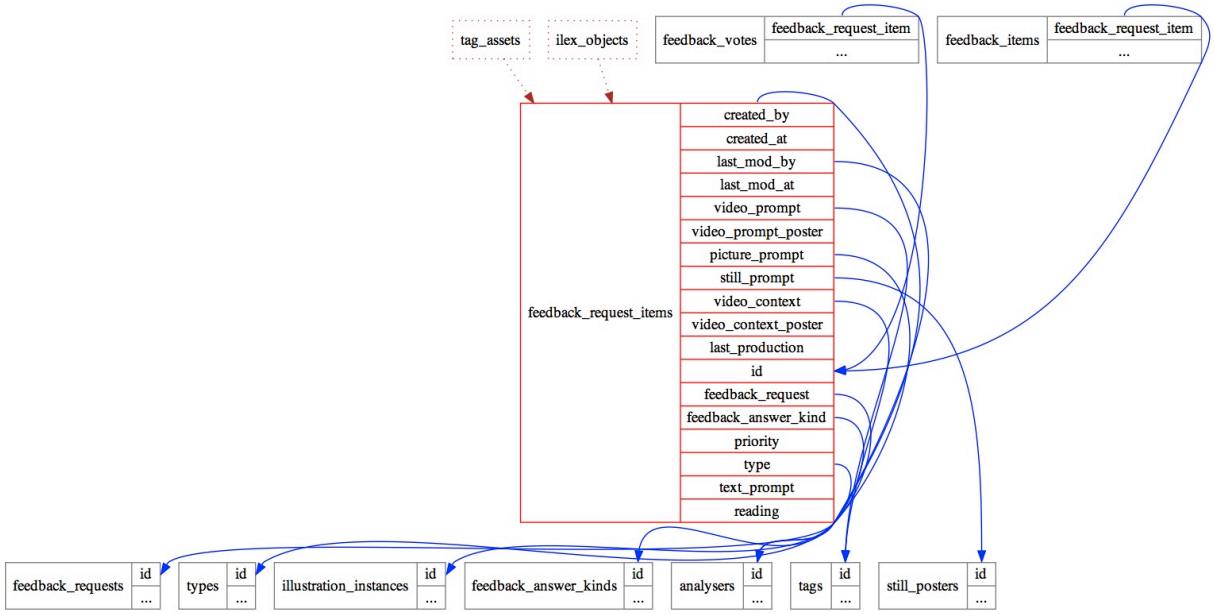


Figure 57: feedback_request_items in the database

What is also recognizable is that metadata entries decribe the creation date, last modification date, authors etc. of database artefacts overall tables.

12.2 Results

On the result side the feedback answers are persisted in the iLex database as well. The process of converting xml data result packidges into SQL statements by the use of XSLT[13] will be part of the following chapter 13.
First of all the *feedback_items* table is considered. This corresponds to the results of the Feedback web-application fetched from the webapp's file-system.
The id column is used for the packidge_id of the specific result document. The Feedback user who has answered the questionnaire is considered as an informant inside the iLex system according to its user concept. The informant becomes explicitly referenced inside the specific *feedback_items* record.

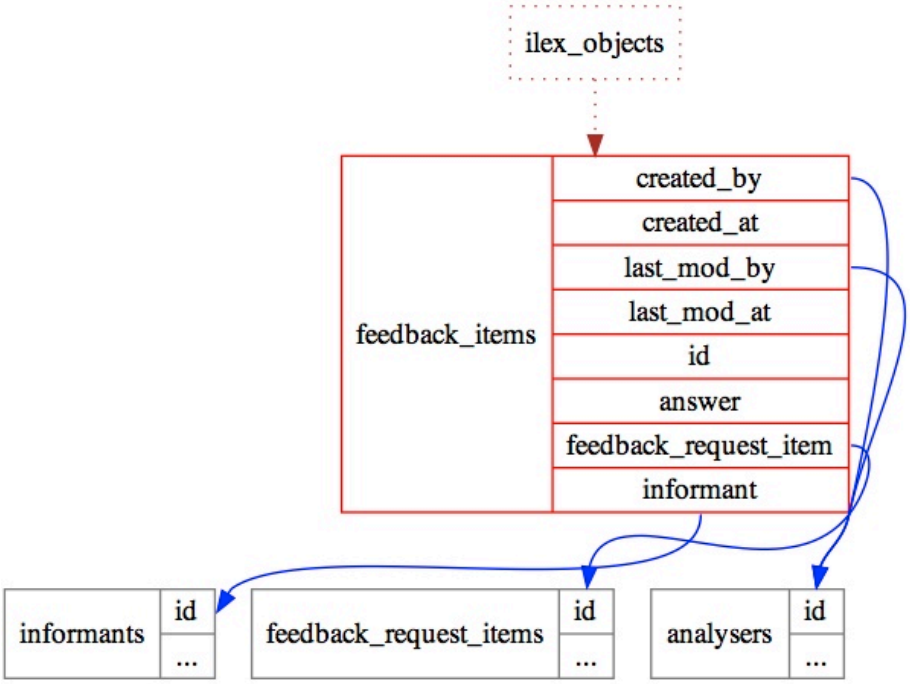


Figure 58: feedback_items as results

[13] Extensible Stylesheet Language Transformations

Since pages are only a container structure inside the packidge xml the row nodes are in focus of the xml parsing and processing in this context because rows contain the real data. Therefore the *feedback_request_item* column resides inside the table for the xml results.
Please note the occurrence of the already mentioned metadata about the creator and modification status here as well.

12.3 State Transitions

feedback_requests entities can be in a number of different states. The following diagram shows all states as well as the possible transitions between these:

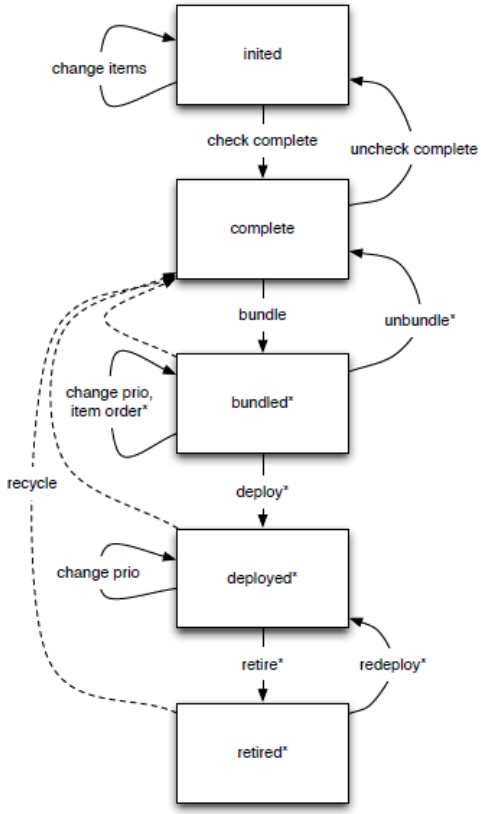


Figure 59: State transitions of feedback_requests

Downwards transitions reflect the regular workflow, upwards transitions are any kind of repair action. Requests in bundles can be recycled (i.e. the system duplicates the requests to be recycled) independent of the state of the bundles.
The states are implemented as virtual columns for both *feedback_bundles* and *feedback_requests* and are stored as:

	feedback_requests.completed	**EXISTS (feedback_deployments.id)**	**feedback_deployments.active**
inited	false	No bundle record exists	
complete	true		
bundled	true	false	n/a
deployed	true	true	At least one is true and is a deployment to a group or has active individual deployments
retired	true	true	All are false and none has individual deployments

Table 1: States overview

For *feedback_deployments* and *feedback_individual_deployments*, the same virtual column exists, but will only return deployed or retired. Compare to figure 60 for deployments and to figure 61 for individual deployments.

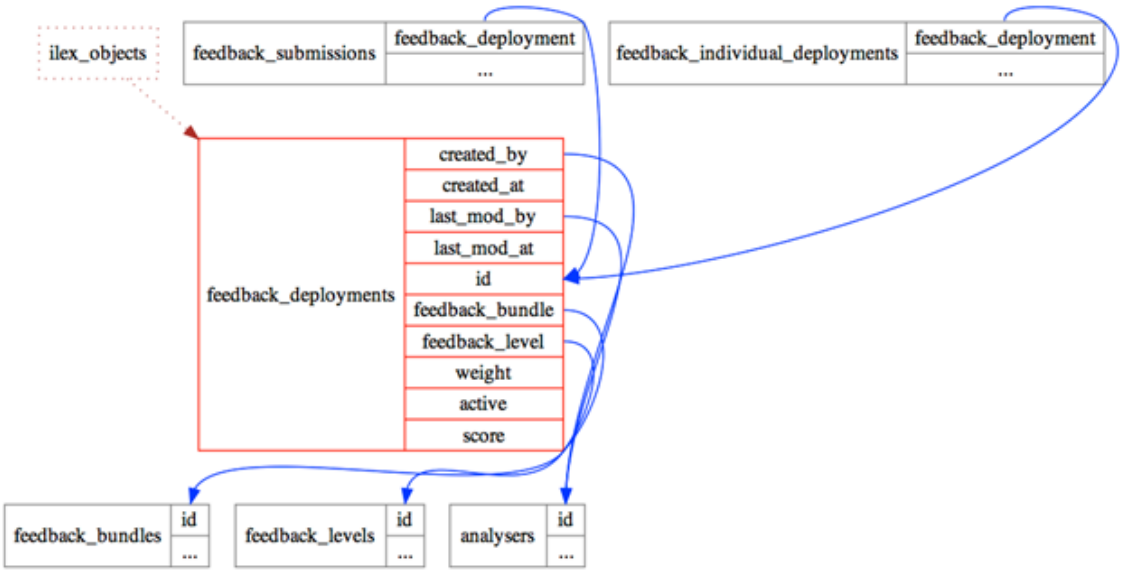


Figure 60: feedback_deployments database relations

The workflow transitions have the following functionality (besides changing the state):

check_complete	The user checkmarks the complete option which is only possible if all necessary fields according to the specification in the *feedback_template* are filled.
bundle	Creates a *feedback_bundles* record to hold all selected *feedback_requests* records. The action allows the user to specify a priority for each category and user this bundle shall be deployed to later.
deploy	Builds the package as well as all related media resources and deploys them to the target system. It then updates the target system's system and/or user files to reflect the new package. While deployed, only the package's priority can be changed.
retire	Removes the package from all system and user files so that it won't be delivered to any more user requesting a new package.

Table 2: Workflow transitions

A deployment to individual users will automatically be marked as inactive once the last user has submitted the bundle. This may result in an automatic update of the bundle's status from deployed to retired.

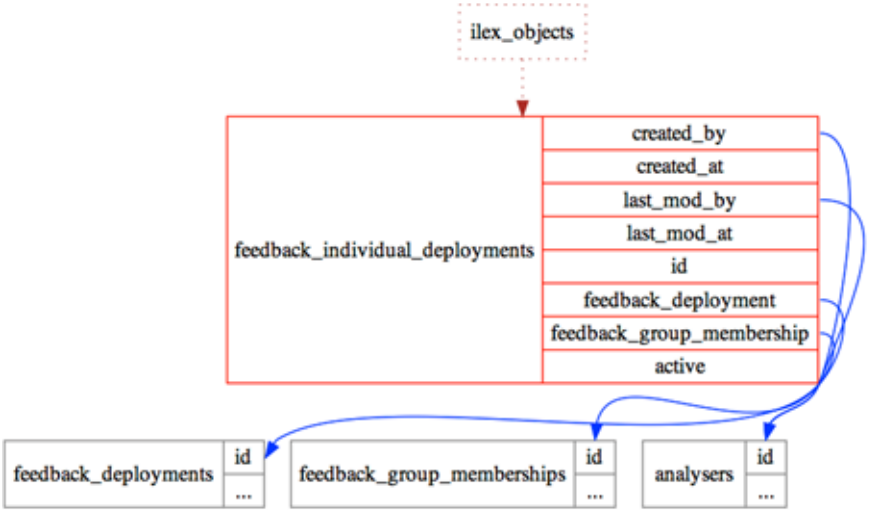


Figure 61: feedback_individual_deployments

Before a *feedback_bundles* record is deployed into a feedback system packidge, additional checking is performed. *feedback_bundles.ready_to_deploy* not only checks the data of the bundle, but also calls *feedback_requests.ready_to_deploy* and *feedback_request_items.ready_to_deploy*. (This checking, however, does not test if the movie files needed to view the packidge have been delivered to the http server.) In addition, a packidge cannot be deployed to a user group more than once, and is not possible if an individual deployment to a member of that group has already happened.

13. The XSL Tranformation Process

As being discussed in the chapter before packidge xml documents are parsed and then processed by XSL Transformations. In this way SQL statements are generated from the relevant xml data entries the user has commited in the results. In this way a loose coupling between the Feedback system and iLex is achieved.
As we have seen in the previous chapters iLex fetches the xml data from the Feedback system's webapp directory namely, the file system.
In this chapter a general reference is given on how the transformation works. In the following chapter the documentation is going to relate to those concepts by showing the real results inside of iLex.
The basic concept is clear: XSL Transformations are operating on the packidge-XML data and result in in SQL statements that are excecuted in order to populate the corresponding database tables we have considered in the previous chapter.
The following example shows how the climbing up and down in the DOM-Tree of an XML packidge result is carried out in general.

13.1 Example XSL

Let's take a look at the following XSL Transformation:

```
<xsl:for-each                          select="//button[@icon='YES'                          and
@selected='true']/../../row/button[@icon='SIGN_KNOWN' and @selected='true']/../@id">
INSERT     INTO     feedback_votes(feedback_request_item,feedback_submission,value)     VALUES
(<xsl:value-of select="." />,$1,'SIGN_KNOWN');
</xsl:for-each>
```

Listing 65: Example XSL Transformation

A packidge xml document is processed inside an xsl for-each loop. The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set.
The value of the *'select'* attribute is an XPATH[14] expression. This kind of expression works like navigating inside a file system where a forward slash (/) selects subdirectories.
In this case a <button> element is selected. The selection matches all buttons because the double slash (//) refers to all occurrences no matter on which position the node is placed.
In other words "//" selects nodes in the document from the current node that match the selection no matter where they are.
The button tags itself have the attributes *'icon'* with the value of 'YES' and at the same time (logical AND) an attribute *'selected'* with the value of 'true'. This states the basic requirement for the selction. Compare to *@icon* and *@selected* in listing 65. This directly refers to the packidge xml attributes.
The following listing shows a matching component inside a packidge xml result.

```
<row id="4066" type="buttons" goto="next-sub-page">
  <content id="4066" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.webm" />
  </content>
  <button icon="YES" selected="true" goto="next-sub-page">Ja</button>
  <button icon="NO" goto="exit-page">Nein</button>
</row>
```

Listing 66: Matching component inside a packidge xml result

By having a further look on the XSL navigation through the XML DOM-Tree the occurrence of ".." shows the selection of the parent of the current node (XPATH).
/../../row/button[@icon='SIGN_KNOWN' and @selected='true']

Navigates up the DOM-Tree to the parent node <page> in order to reach a row node again afterwards that contains another button-construct. This is characterized by an XPATH expression in square brackets.
Compare to the next listing.

```
<row id="4002" type="buttons" goto="next-sub-page">
  <content id="4002" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1174/4002/2300881.jpg" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1174/4002/2300881.m3u8" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1174/4002/2300881.mp4" />
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1174/4002/2300881.webm" />
  </content>
  <content id="content1" hratio="360" vratio="270" type="keyword">Hochzeit (Fest)</content>
  <button icon="SIGN_USED" goto="next-sub-page">Benutze ich</button>
  <button icon="SIGN_KNOWN" selected="true" goto="next-sub-page">Kenne ich</button>
  <button icon="SIGN_UNKNOWN" goto="next-sub-page">Kenne ich nicht</button>
</row>
```

Listing 67: button-construct

[14] http://en.wikipedia.org/wiki/XPath

In this case only constructs are considered that contain a button icon with value “SIGN_KNOWN”. Furthermore the attribute *‘selected’* has to be available and containing the value of ‘true’.
Where exactly one is located inside the DOM becomes clear by the evaluation of the corresponding row id (/../@id via the parent node) for the buttons.
Compare to the listing above: row id=“4002“ inside packidge 75 refering to the red marked button.

13.2 Generation of SQL statements for iLex

The following SQL statement is included inside the xsl for each element

```
INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES
(<xsl:value-of select="." />,$1,'SIGN_KNOWN');
```

Only these columns are affected inside the feedback_votes table.
The <xsl:value-of> element extracts the value of a selected node.
The <xsl:value-of> element can also be used to select the value of an XML element and adds it to the output.
This is inserted into the feedback_request_item column. Compare the next figure.
The <xsl:value-of select="."/> puts out the current tag content.
In order to make the concept more demonstrative let us have a look at the corresponding table in this context: *feedback_votes*.
The *fedback_request_item* value inside *feedback_votes* record points to a certain *feedback_request_item* id which corresponds to a row id inside the packidge xml.

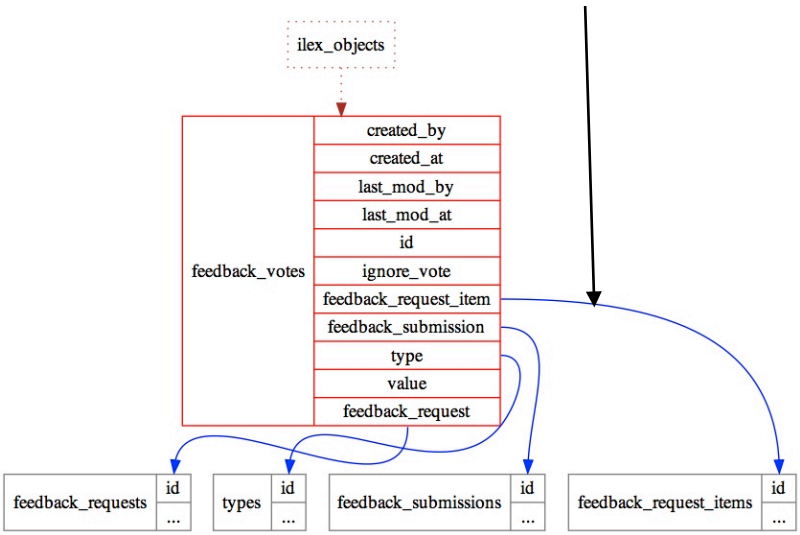


Figure 62: feedback_votes

The following figure illustrates the **affected columns** by the SQL statement that has been discussed.

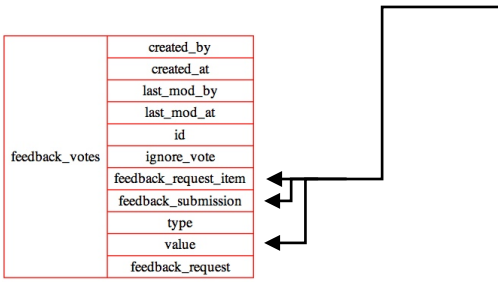


Figure 63: Affected columns in feedback_votes

The resulting feedback entry of the user - considered as a submission such as “Kenne ich” or “Ja” is handled by the column *feedback_submission*. This relates to the *feedback_submissions* table that represents the user’s submissions by addressing them with a unique id as can be seen in the following figure.

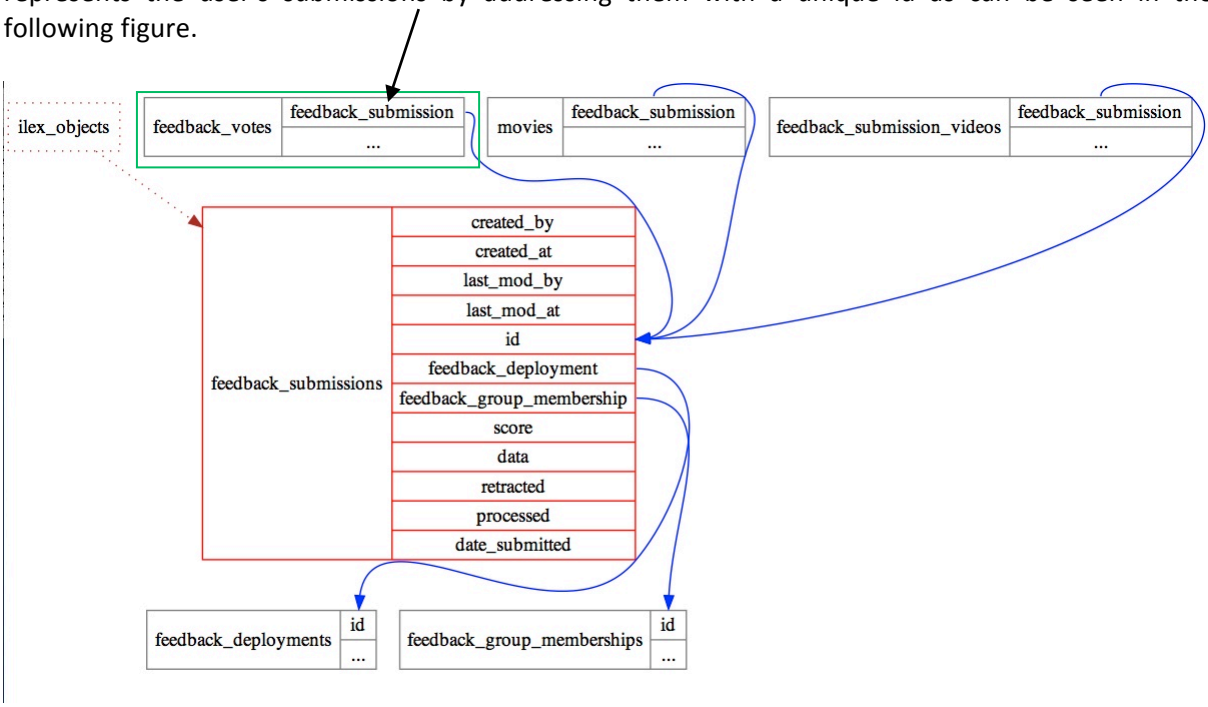
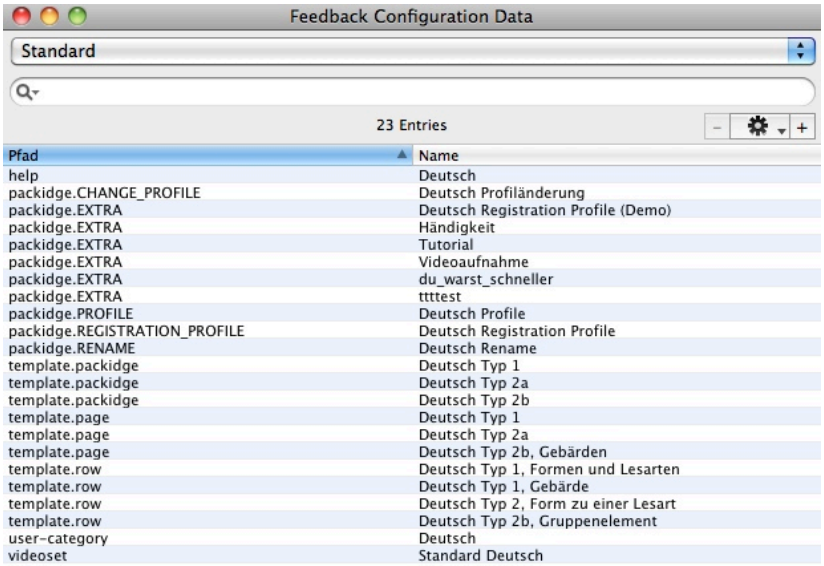


Figure 64: feedback_submissions table

The “value” column in *feedback_votes* is represented by the ‘SIGN_KNOWN’ literal in the SQL statement.

14. Feedback Configuration Data

The Feedback web-application is provided with new xml data by the iLex transcription environment regularly. The task “(Automatically) Deploy Feedback” is responsible for the deployment of the generated xml files inside Feedback’s webapps/packidge directory. This task can be excecuted manually or automatically.
In order to get the two applications communicating with each other the XML standard seems to be the appropriate solution on technical level. To make the creation and handling of questionnaires as comfortable as possible iLex contains many XML templates for the creation of Feedback packidges. These templates come in different flavours depending on their later purpose inside of Feedback.
The templates can be basically distinguished into dynamical templates that are modularly and hierarchically filled with content by the staff and static templates which have a fixed content.
Dynamic templates are marked with the prefix “template.” in iLex whereas static templates contain the prefix ”packidge.” such as “packidge.EXTRA”.
The feedback_configurations are diagramed in iLex as follows:



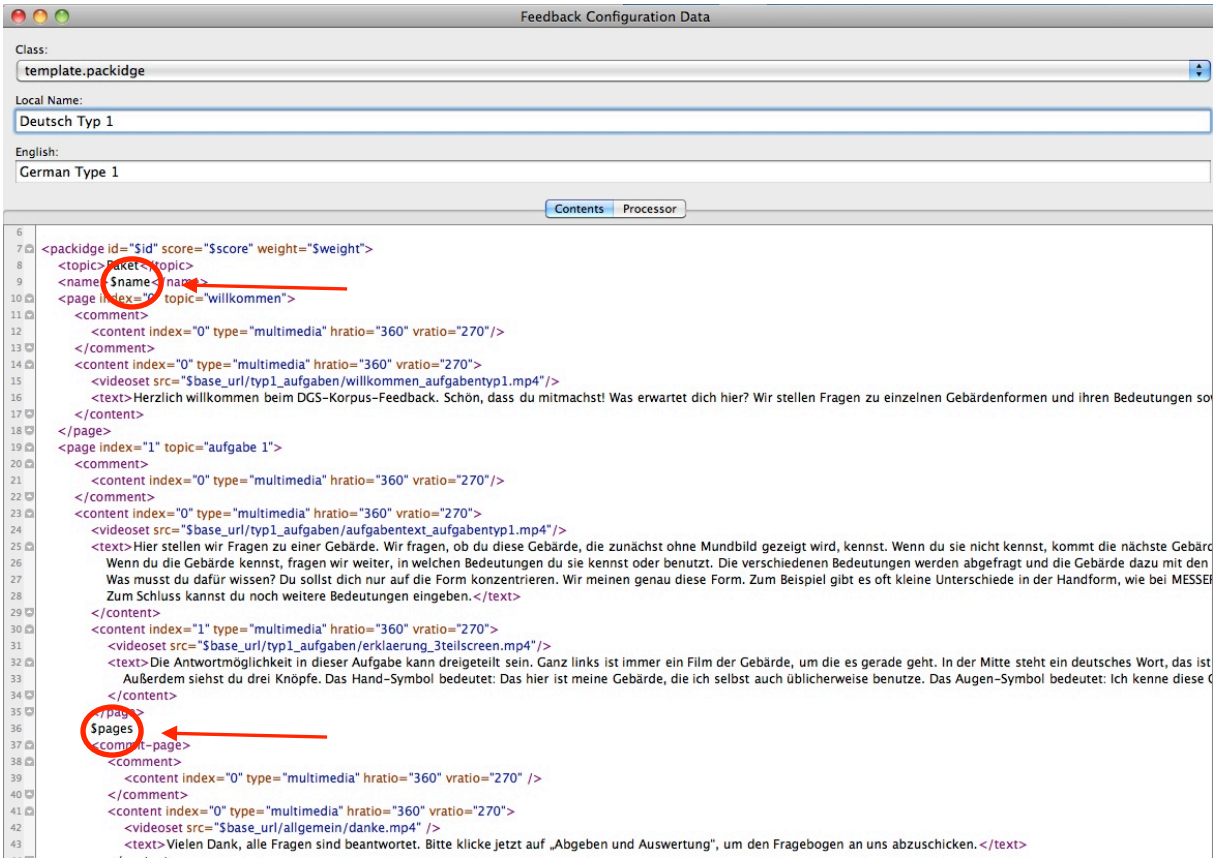
Pfad	Name
help	Deutsch
packidge.CHANGE_PROFILE	Deutsch Profiländerung
packidge.EXTRA	Deutsch Registration Profile (Demo)
packidge.EXTRA	Händigkeit
packidge.EXTRA	Tutorial
packidge.EXTRA	Videoaufnahme
packidge.EXTRA	du_warst_schneller
packidge.EXTRA	ttttest
packidge.PROFILE	Deutsch Profile
packidge.REGISTRATION_PROFILE	Deutsch Registration Profile
packidge.RENAME	Deutsch Rename
template.packidge	Deutsch Typ 1
template.packidge	Deutsch Typ 2a
template.packidge	Deutsch Typ 2b
template.page	Deutsch Typ 1
template.page	Deutsch Typ 2a
template.page	Deutsch Typ 2b, Gebärden
template.row	Deutsch Typ 1, Formen und Lesarten
template.row	Deutsch Typ 1, Gebärde
template.row	Deutsch Typ 2, Form zu einer Lesart
template.row	Deutsch Typ 2b, Gruppenelement
user-category	Deutsch
videoset	Standard Deutsch

Figure 65: Feedback Configuration Data (feedback_configurations)

The purpose is to enable the staff to create packages without having knowledge about XML itself.
An employee is able to create packidges from a ready-to-go feedback_configuration. This refers to the mentioned static packidges that are characterized by hardwired, static data that don't has to be provided with dynamic components like pages, rows, assets etc. Compare to the representation of the packidge.EXTRA elements.
The "normal" packidges whereas are bundled from Feedback-Requests that are an entity component in iLex itself. Compare to chapter 20.2 "Bundling questions as a package". Packidges with the „template."-prefix (in the preceding figure) become populated in that way. These templates are nested hierarchically (cmp. Figure 53). A questionanire template contains pages (templates), pages contain rows (templates) and rows the real content.
Let us have a deeper insight into the Feedback templates at this point since this represents a central concept of the data interchange from iLex to Feedback.

14.1 Package Templates

A template.packidge that can be seen in the following figure is an instance of a special configuration class. These classes are considered later on (chapter 15). The name of the template package refers to the *feedback_proto_bundle* construct (cmp. chapter 16) that determines an assignment to certain content types in the DGS context (Type 1, Type 2a, Type 2b).



Listing 68: template.packidge – German Type 1

In the listing above we can see the familiar xml packidge structure as desribed in chapter 7.1 before. In this case some variables i.e. for “$name” and “$pages” are defined in order to provide them with content dynamically from the questionnaire creation process of an employee. The $pages variable has to be provided with the appropriate page template (template.page). Page templates in turn contain $rows variables that are provided with template.rows of the appropriate type. More on that later on.
Initially let us bring the focus of the “Processor” tab in the iLex GUI. This gives us access to the xsl transformations targeted on the evaluation of the feedback-result-xml-files of the corresponding type.
As described in chapter 13 the XSL Transformations are generating the SQL INSERT statement and therefore are the interface between the two applications.
Since Feedback is not database-related a transformation of the data has to be done. Compare to the following figure:

Feedback Configuration Data

Class:
template.packidge

Local Name:
Deutsch Typ 1

English:
German Type 1

Contents | Processor

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" version="1.0" encoding="UTF-8" indent="no" />
   <xsl:template match="/">
      <!-- implicit unknowns -->
      <xsl:for-each select="//button[@icon='NO' and @selected='true' and @goto='exit-page']/../button[@icon='YES']/../../row/button[@icon='SIGN_UNKNOWN']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_UNKNOWN');
      </xsl:for-each>
      <!-- explicit unknowns -->
      <xsl:for-each select="//button[@icon='YES' and @selected='true']/../../row/button[@icon='SIGN_UNKNOWN' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_UNKNOWN');
      </xsl:for-each>
      <!-- explicit knowns -->
      <xsl:for-each select="//button[@icon='YES' and @selected='true']/../../row/button[@icon='SIGN_KNOWN' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_KNOWN');
      </xsl:for-each>
      <!-- explicit useds -->
      <xsl:for-each select="//button[@icon='YES' and @selected='true']/../../row/button[@icon='SIGN_USED' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_USED');
      </xsl:for-each>
      <!-- recorded new sign, path of feedback_submission_video does not contain file name extension -->
      <xsl:for-each select="//button[@icon='VIDEO_TEXT' and @selected='true']/../../row[@type='video-and-text']/content/video/@src">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,'SIGN_USED: <xsl:value-of select="." />');
      </xsl:for-each>
      <!-- described new sign -->
      <xsl:for-each select="//button[@icon='VIDEO_TEXT' and @selected='true']/../../row[@type='video-and-text']/content/text">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$SIGN_USED: <xsl:value-of select="." />$$);
      </xsl:for-each>
      <!-- video comments on contents pages -->
      <xsl:for-each select="//page[number(@id)=@id]/comment/content/video/@src">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,'COMMENT-VIDEO: <xsl:value-of select="." />');
      </xsl:for-each>
      <!-- text comments on contents pages -->
      <xsl:for-each select="//page[number(@id)=@id]/comment/content/text">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$COMMENT-TEXT: <xsl:value-of select="translate(.,'&#xD;&#xA;',' ')" />$$);
      </xsl:for-each>
      <!-- video comments on general pages -->
      <xsl:for-each select="//page[number(@id)!=number(@id)]/comment/content/video/@src">
         INSERT INTO feedback_votes(feedback_submission,value) VALUES ($1,'COMMENT-VIDEO: <xsl:value-of select="." />');
      </xsl:for-each>
```

Listing 69: Processor XSLT / Building of SQL statements

Other questionnaire types are derived from other templates such as the template.packidge “German Type 2a”.
Technically there is no difference between these types. But regarding the content of the packidges a distinction has to be made. So a different template comes into play.

Feedback Configuration Data

Class:
template.packidge
Local Name:
Deutsch Typ 2a
English:
German Type 2a

Contents | Processor

```
<packidge id="$id" score="$score" weight="$weight">
   <topic>Paket</topic>
   <name>$name</name>

   <page index="0" topic="willkommen">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/typ2_aufgaben/typ2_willkommen.mp4"/>
         <text>Hallo! Hier in diesem Aufgabenpaket geht es um regionale Unterschiede. Wie du weißt, gibt es in verschiedenen Regionen Deutschlands unterschiedliche Gebärd
      </content>
   </page>

   <page index="1" topic="aufgabe">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/typ2_aufgaben/typ2_aufgabe.mp4"/>
         <text>Bei dieser Frage konzentrieren wir uns immer auf eine einzelne Bedeutung. Es ist möglich, dass eine Gebärde auch noch andere Bedeutungen hat, aber im Mom
      </content>
   </page>

   $pages

   <commit-page>
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270" />
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/allgemein/danke.mp4" />
         <text>Vielen Dank, alle Fragen sind beantwortet. Bitte klicke jetzt auf „Abgeben und Auswertung", um den Fragebogen an uns abzuschicken.</text>
      </content>
   </commit-page>
   <revise-page>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/allgemein/revise.mp4" />
         <text>Du bist am Ende des Fragebogens angekommen, aber es gibt Fragen, die du noch nicht beantwortet hast. Bitte gehe zurück zur ersten offenen Frage.</text>
      </content>
```

Listing 70: Template.packidge - German Type 2a

The XSLT is applied to „Typ2a“ result packidges in the same way as “Type 1” is handled by iLex.

Feedback Configuration Data

Class:
template.packidge

Local Name:
Deutsch Typ 2a

English:
German Type 2a

Contents | Processor

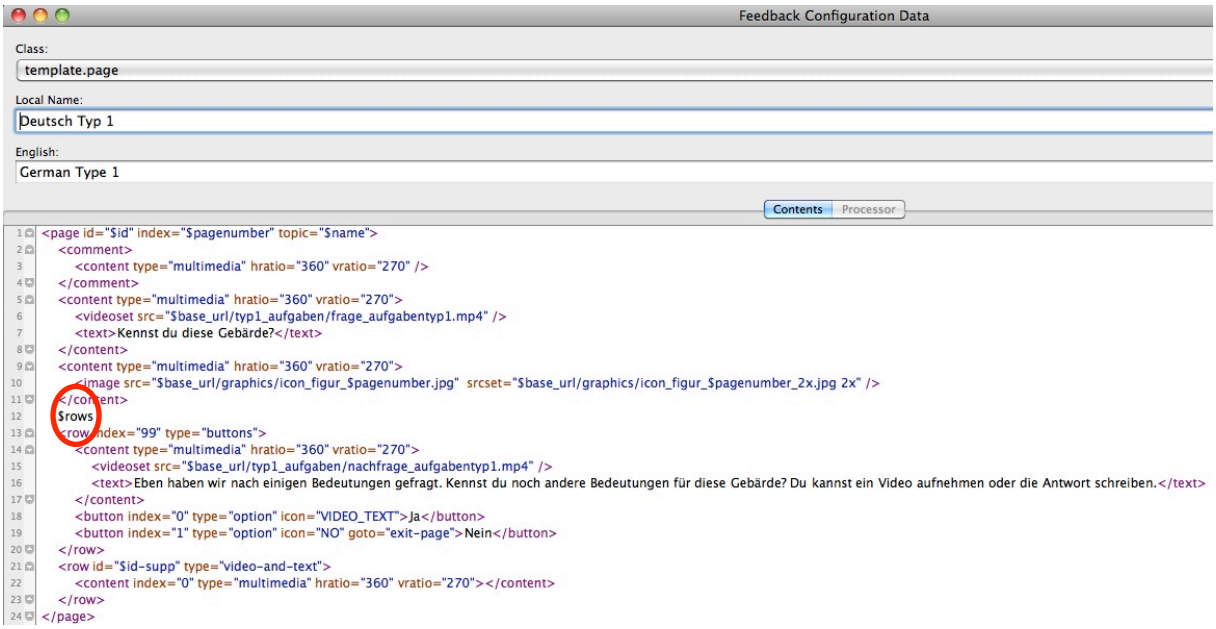
```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" version="1.0" encoding="UTF-8" indent="no" />
   <xsl:template match="/">
      <!-- explicit unknowns -->
      <xsl:for-each select="//row/button[@icon='SIGN_UNKNOWN' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_UNKNOWN');
      </xsl:for-each>
      <!-- explicit knowns -->
      <xsl:for-each select="//row/button[@icon='SIGN_KNOWN' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_KNOWN');
      </xsl:for-each>
      <!-- explicit knowns -->
      <xsl:for-each select="///row/button[@icon='SIGN_USED' and @selected='true']/../@id">
         INSERT INTO feedback_votes(feedback_request_item,feedback_submission,value) VALUES (<xsl:value-of select="." />,$1,'SIGN_USED');
      </xsl:for-each>
      <!-- recorded new sign, path of feedback_submission_video does not contain file name extension -->
      <xsl:for-each select="//button[@icon='VIDEO_TEXT' and @selected='true']/../../row[@type='video-and-text']/content/video/@src">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$SIGN_USED: <xsl:value-of select="." />$$);
      </xsl:for-each>
      <!-- described new sign -->
      <xsl:for-each select="//button[@icon='VIDEO_TEXT' and @selected='true']/../../row[@type='video-and-text']/content/text">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$SIGN_USED: <xsl:value-of select="translate(.,'$','§')" />$$);
      </xsl:for-each>
      <!-- video comments on contents pages -->
      <xsl:for-each select="//page[number(@id)=@id]/comment/content/video/@src">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$COMMENT-VIDEO: <xsl:value-of select="." />$$);
      </xsl:for-each>
      <!-- text comments on contents pages -->
      <xsl:for-each select="//page[number(@id)=@id]/comment/content/text">
         INSERT INTO feedback_votes(feedback_request,feedback_submission,value) VALUES (<xsl:value-of select="../../../../@id" />,$1,$$COMMENT-TEXT: <xsl:value-of select="translate(.,'&#xD;&#xA;','§  ')" />$$);
      </xsl:for-each>
      <!-- video comments on general pages -->
      <xsl:for-each select="//page[number(@id)!=number(@id)]/comment/content/video/@src">
         INSERT INTO feedback_votes(feedback_submission,value) VALUES ($1,$$COMMENT-VIDEO: <xsl:value-of select="." />$$);
      </xsl:for-each>
      <!-- text comments on general pages -->
      <xsl:for-each select="//page[number(@id)!=number(@id)]/comment/content/text">
         INSERT INTO feedback_votes(feedback_submission,value) VALUES ($1,$$COMMENT-TEXT: <xsl:value-of select="translate(.,'$&#xD;&#xA;','§  ')" />$$);
      </xsl:for-each>
   </xsl:template>
</xsl:stylesheet>
```

Listing 71: The corresponding XSLT

The third type is the template.packidge “Type 2b”. In relation to its content it is characterized by regional differences in the use of the sign language. Technically it makes no difference for the interface between the two systems and is handled in the identical way described.
In order to create template packages select the following path:
Data --> Parameters --> Feedback Configuration Data --> +
Then select a class (for special packages - i.e. info package, special requests - use packidge.EXTRA or create a new template under Feedback Configuration Classes)

14.2 Page Templates

For the three different types of template.packidges there exist corresponding page templates (template.page).
This chapter shows how the nesting of content is carried out on this level in order to prevent staff from working with xml directly.



Listing 72: Type 1

In each type of page templates a $rows variable can be found which acts as an access point for inserting the row contents containing the payload data.

Feedback Configuration Data

Class:
template.page

Local Name:
Deutsch Typ 2a

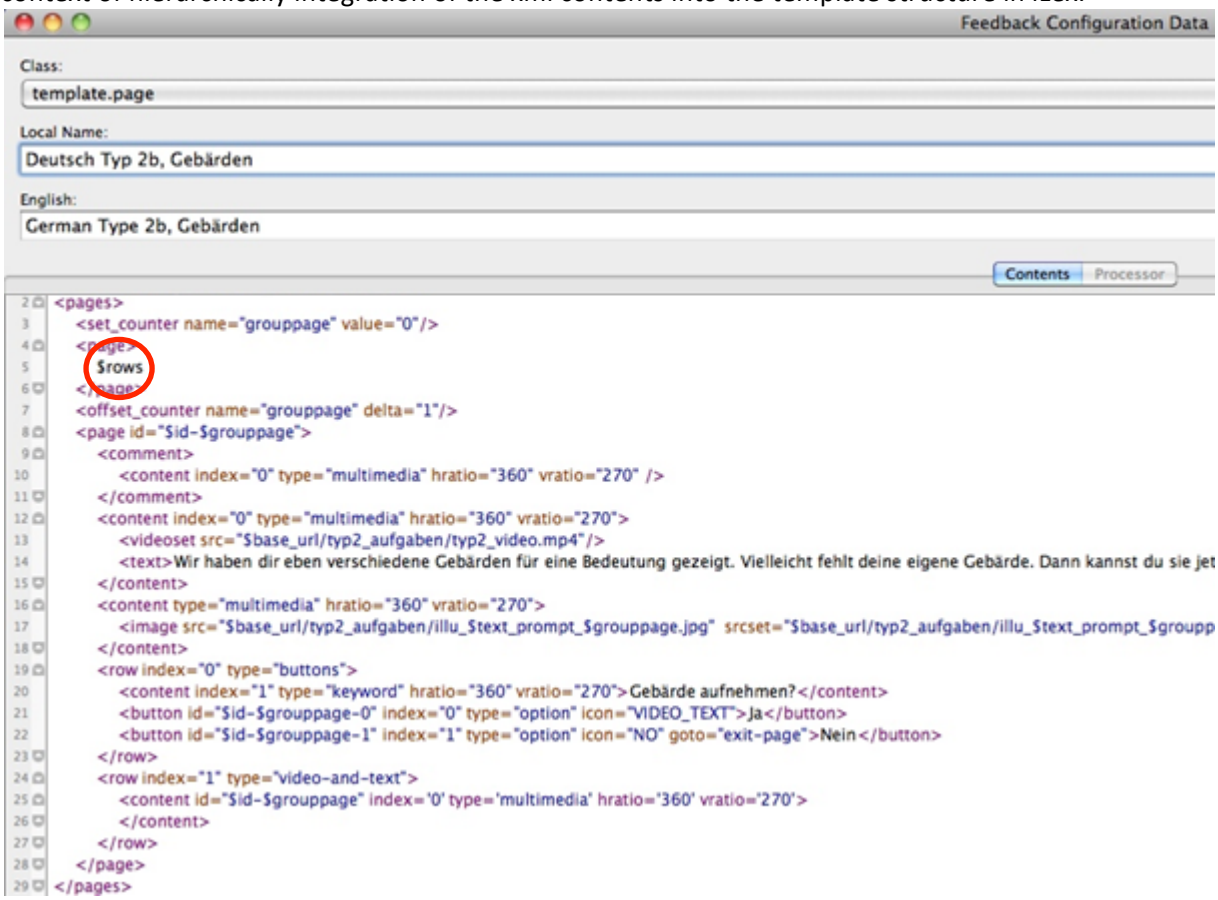
English:
Deutsch Typ 2a

Contents | Processor

```
<page id="$id" index="$pagenumber" topic="$name">
    <comment>
        <content type="multimedia" hratio="360" vratio="270" />
    </comment>
    <content type="multimedia" hratio="360" vratio="270">
        <videoset src="$base_url/typ2_aufgaben/typ2_frage.mp4" />
        <text>Kennst du oder benutzt du die gezeigte Gebärde für die angegebene Bedeutung? Oder ist sie dir unbekannt?</text>
    </content>
    <content type="multimedia" hratio="360" vratio="270">
        <image src="$base_url/typ2_aufgaben/illu_$text_prompt.jpg"  srcset="$base_url/typ2_aufgaben/illu_$text_prompt_2x.jpg 2x" />
    </content>

    $rows

    <row index="99" type="buttons">
        <content type="multimedia" hratio="360" vratio="270">
            <videoset src="$base_url/typ2_aufgaben/typ2_video.mp4" />
            <text>Wir haben dir eben verschiedene Gebärden für eine Bedeutung gezeigt. Vielleicht fehlt deine eigene Gebärde. Dann kannst du sie jet
        </content>
        <button index="0" type="option" icon="VIDEO_TEXT">Ja</button>
        <button index="1" type="option" icon="NO" goto="exit-page">Nein</button>
    </row>
    <row id="$id-supp" type="video-and-text">
        <content index="0" type="multimedia" hratio="360" vratio="270"></content>
    </row>
</page>
```

Listing 73: Type 2a

The figure above (Type 2b) as well as the figure below (Type 2b) both show the same structure and context of hierarchically integration of the xml contents into the template structure in iLex.




```
<pages>
   <set_counter name="grouppage" value="0"/>
   <page>
      $rows
   </page>
   <offset_counter name="grouppage" delta="1"/>
   <page id="$id-$grouppage">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270" />
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/typ2_aufgaben/typ2_video.mp4"/>
         <text>Wir haben dir eben verschiedene Gebärden für eine Bedeutung gezeigt. Vielleicht fehlt deine eigene Gebärde. Dann kannst du sie jet
      </content>
      <content type="multimedia" hratio="360" vratio="270">
         <image src="$base_url/typ2_aufgaben/illu_$text_prompt_$grouppage.jpg"  srcset="$base_url/typ2_aufgaben/illu_$text_prompt_$groupp
      </content>
      <row index="0" type="buttons">
         <content index="1" type="keyword" hratio="360" vratio="270">Gebärde aufnehmen?</content>
         <button id="$id-$grouppage-0" index="0" type="option" icon="VIDEO_TEXT">Ja</button>
         <button id="$id-$grouppage-1" index="1" type="option" icon="NO" goto="exit-page">Nein</button>
      </row>
      <row index="1" type="video-and-text">
         <content id="$id-$grouppage" index='0' type='multimedia' hratio='360' vratio='270'>
         </content>
      </row>
   </page>
</pages>
```

Listing 74: Type 2b

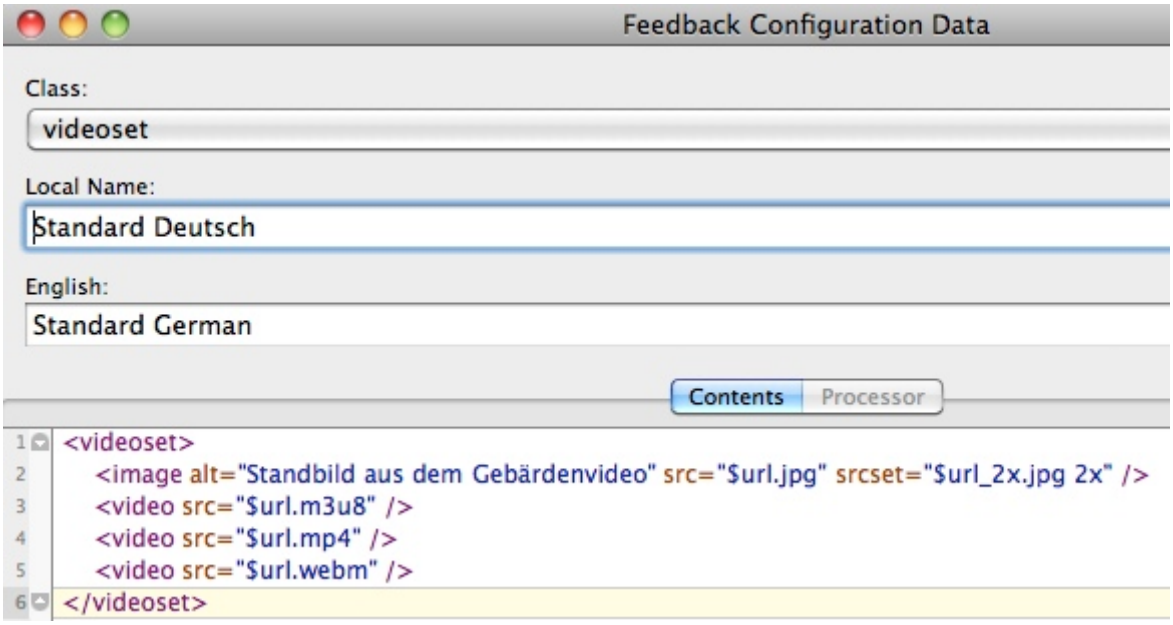
14.3 Row Templates

The row templates contain inter alia the answer button structure that has been discussed earlier and represents the target sructure for videoset templates that are provided with web-appropriate video streams.
The different template structures enable the webapp to give some additional information on a special use of a sign for example.

```
<row id="4067" type="buttons">
    <content id="4067" index="2" type="multimedia" hratio="360" vratio="270">

        <image  alt="Standbild aus dem Gebärdenvideo"
                src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.jpg"
                srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380_2x.jpg 2x"
        />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.webm" />

    </content>
    <content index="1" type="keyword" hratio="360" vratio="270">Tier</content>

    <button id="4067-2" index="0" type="option" icon="SIGN_USED">Benutze ich</button>
    <button id="4067-1" index="1" type="option" icon="SIGN_KNOWN">Kenne ich</button>
    <button id="4067-0" index="2" type="option" icon="SIGN_UNKNOWN">Kenne ich nicht</button>
</row>
```

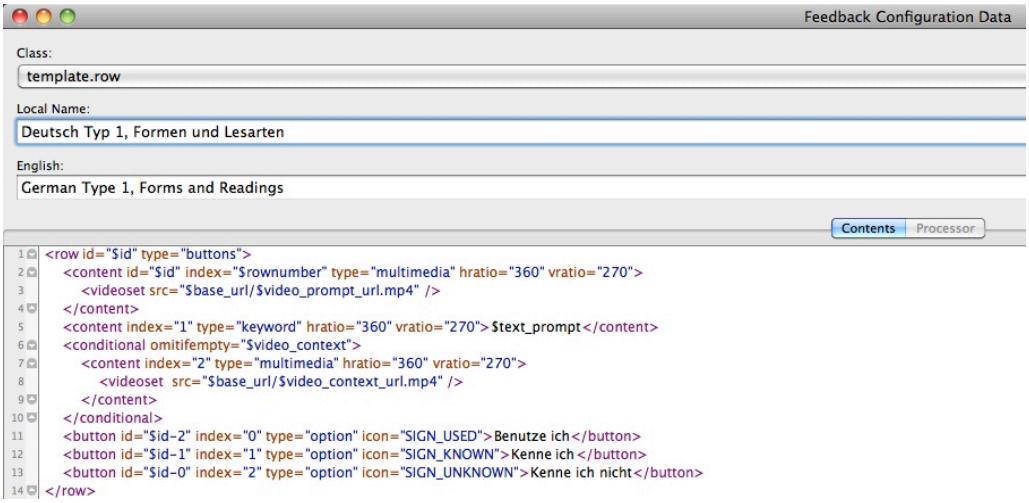
Listing 75: Type 1 xml listing

This part (arrow) became generated from the videoset template. When the process “Deploy Feedback “ is applied some video streaming background jobs are executed such as rendering to an appropriate web based format. The real path urls are injected by the videoset template.



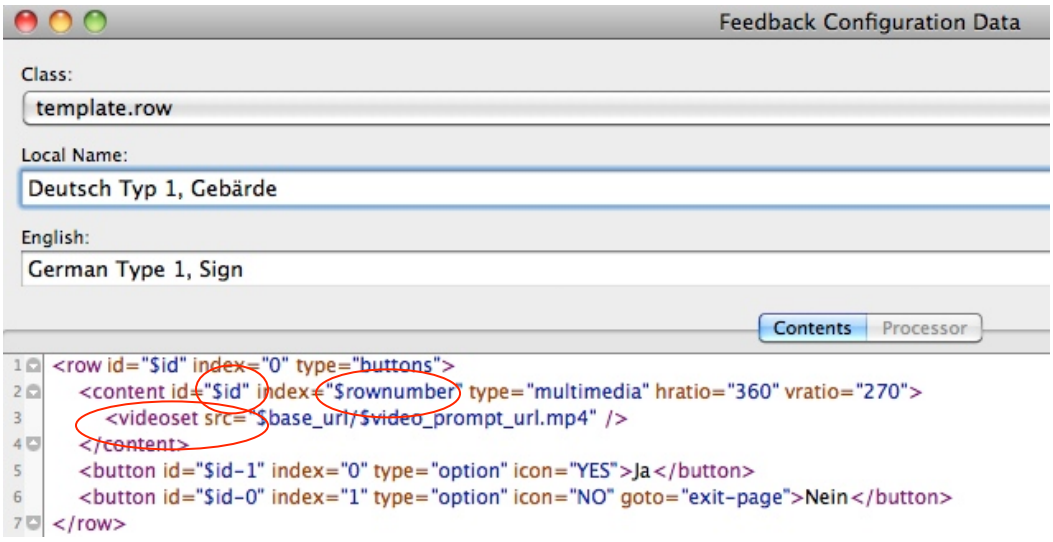
Listing 76: Feedback class videoset

Having a look at the next listing of the template.row “German Type 1, Forms and Readings” one can discover the relations to the xml listing 76 above as follows:
$id of listing 77 becomes provided by the context.
$text_prompt: is called "Dargestellter Text" in the UI and will be provided with literal text for the current sign such as „Tier“ for page topic="123: TIER4-$SAM". Compare the next listing 77. <videoset> correlates to the xml in listing 76.



Listing 77: $variables in an xml template

As an example for “German Type 1, Sign “ we consider the following row template construct.



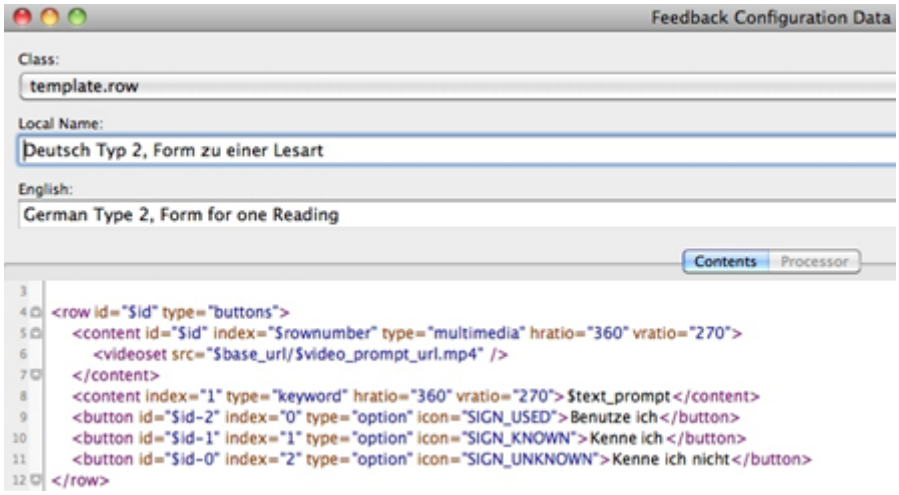
Listing 78: German Type 1, Sign

And its XML representation in the final xml package which makes their relation more obvious.

```
<row id="4066" index="0" type="buttons">
<content id="4066" index="1" type="multimedia" hratio="360" vratio="270">

<image  alt="Standbild aus dem Gebärdenvideo"
        src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.jpg"
        srcset="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378_2x.jpg 2x" />
<video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.m3u8" />
<video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.mp4" />
<video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4066/2303378.webm" />

</content>
<button id="4066-1" index="0" type="option" icon="YES">Ja</button>
<button id="4066-0" index="1" type="option" icon="NO" goto="exit-page">Nein</button>
</row>
```

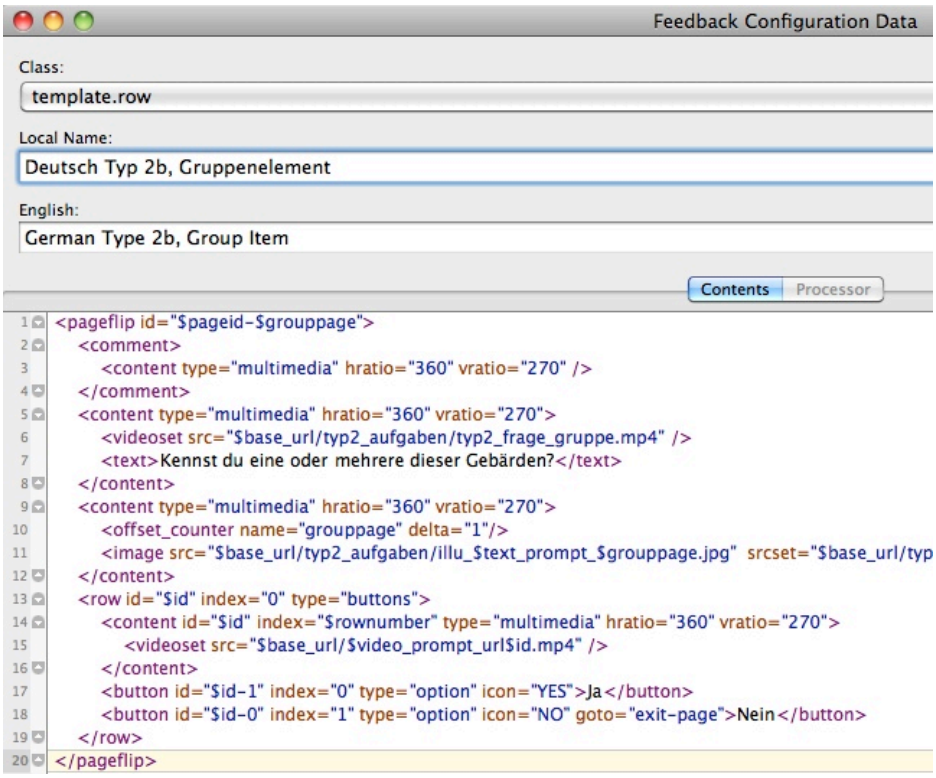
Listing 79: XML representation in the final xml package

These button constructs for example represent the real data which is used by xsl when an answered questionnaire is returned to iLex by a user commitment.
To make the picture complete, let us have a look at the two remaining types of row templates which are “German Type 2, Form for one Reading” and “German Type 2b, Group Item”.



Listing 80: German Type 2, Form for one Reading

Although each template is different from a content related point of view they follow the same technical conception as their predecessors.

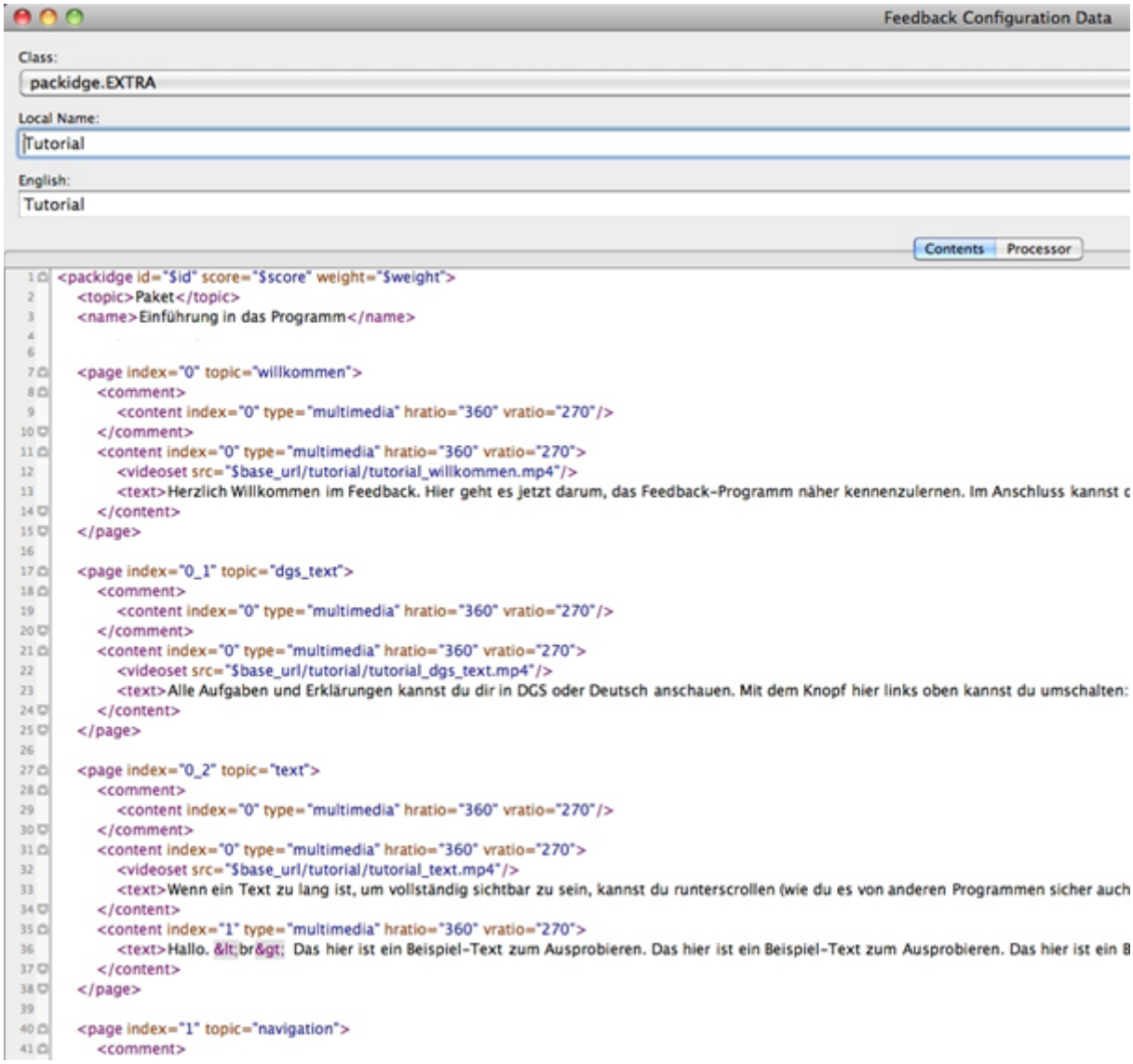



```
<pageflip id="$pageid-$grouppage">
   <comment>
      <content type="multimedia" hratio="360" vratio="270" />
   </comment>
   <content type="multimedia" hratio="360" vratio="270">
      <videoset src="$base_url/typ2_aufgaben/typ2_frage_gruppe.mp4" />
      <text>Kennst du eine oder mehrere dieser Gebärden?</text>
   </content>
   <content type="multimedia" hratio="360" vratio="270">
      <offset_counter name="grouppage" delta="1"/>
      <image src="$base_url/typ2_aufgaben/illu_$text_prompt_$grouppage.jpg"  srcset="$base_url/typ
   </content>
   <row id="$id" index="0" type="buttons">
      <content id="$id" index="$rownumber" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/$video_prompt_url$id.mp4" />
      </content>
      <button id="$id-1" index="0" type="option" icon="YES">Ja</button>
      <button id="$id-0" index="1" type="option" icon="NO" goto="exit-page">Nein</button>
   </row>
</pageflip>
```

Listing 81: German Type 2b, Group Item

14.4 Static Templates

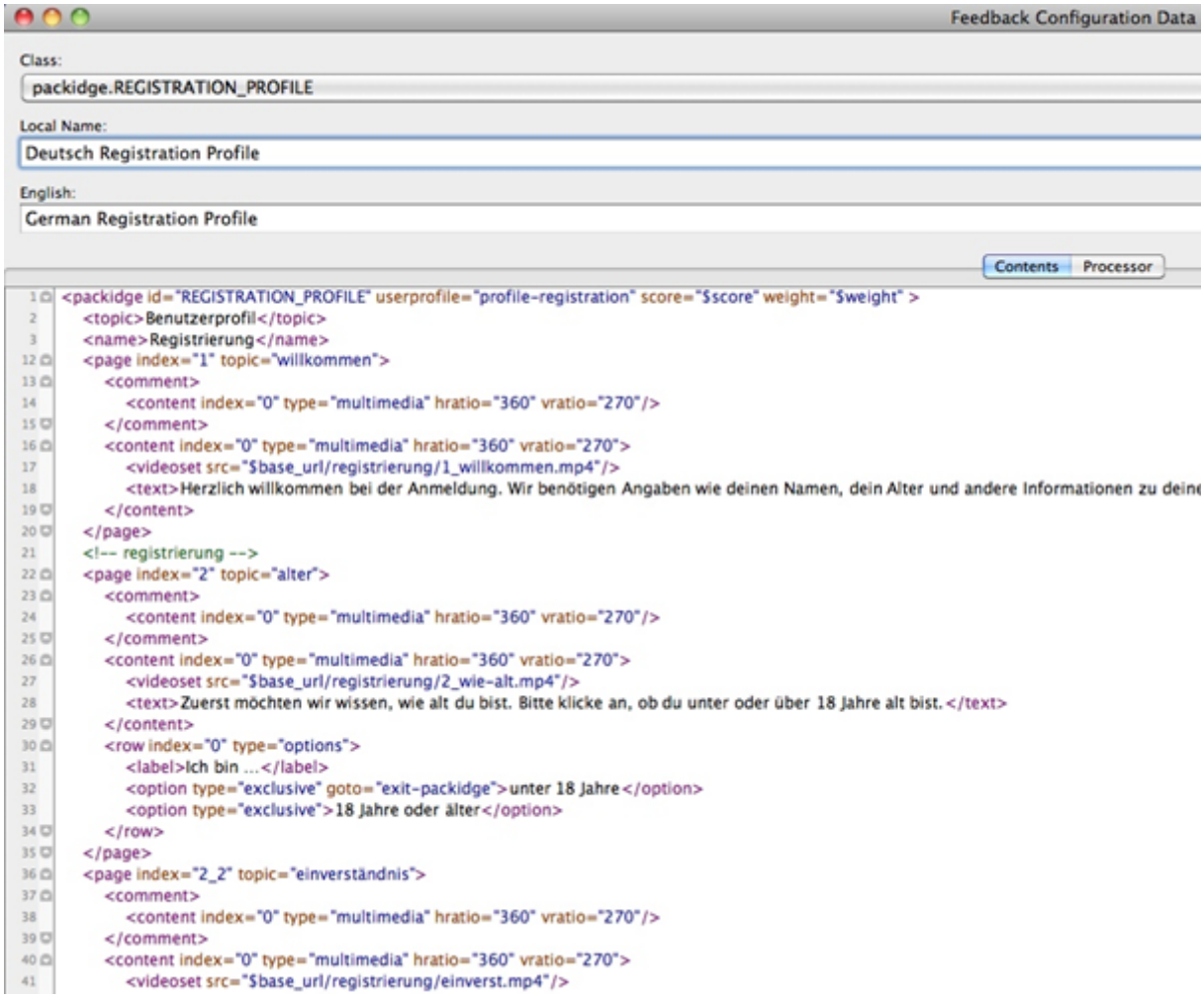
We only have considered dynamic templates so far but as already mentioned there are also static templates in use that are representing metadata questionnaires inside the Feedback system (cf. metadata-related questionnaires in chapter 8.3).
First of all let's have a look at the static packidge.EXTRA templates that are sent out to the Feedback system without adapting their contents. In this case the Feedback-Tutorial package is considered.




```
<packidge id="$id" score="$score" weight="$weight">
   <topic>Paket</topic>
   <name>Einführung in das Programm</name>

   <page index="0" topic="willkommen">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/tutorial/tutorial_willkommen.mp4"/>
         <text>Herzlich Willkommen im Feedback. Hier geht es jetzt darum, das Feedback-Programm näher kennenzulernen. Im Anschluss kannst d
      </content>
   </page>

   <page index="0_1" topic="dgs_text">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/tutorial/tutorial_dgs_text.mp4"/>
         <text>Alle Aufgaben und Erklärungen kannst du dir in DGS oder Deutsch anschauen. Mit dem Knopf hier links oben kannst du umschalten:
      </content>
   </page>

   <page index="0_2" topic="text">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/tutorial/tutorial_text.mp4"/>
         <text>Wenn ein Text zu lang ist, um vollständig sichtbar zu sein, kannst du runterscrollen (wie du es von anderen Programmen sicher auch
      </content>
      <content index="1" type="multimedia" hratio="360" vratio="270">
         <text>Hallo. &lt;br&gt; Das hier ist ein Beispiel-Text zum Ausprobieren. Das hier ist ein Beispiel-Text zum Ausprobieren. Das hier ist ein B
      </content>
   </page>

   <page index="1" topic="navigation">
      <comment>
```

Listing 82: The static packidge.EXTRA template

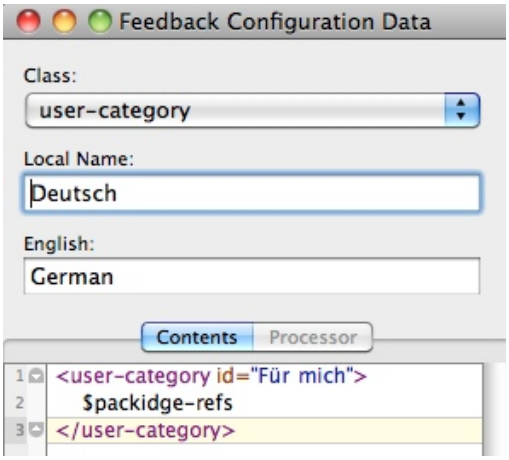
As we become aware the <page></page> node constructs are already contained in this structure. They do not have to be filled with content since they already are ready to be sent out to Feedback. For technical reasons there are also some variables inside this structure because this is a template. However these variables are not content related which is a big difference to dynamically filled templates we already considered.
The same issue we can consider with the **REGISTRATION_PROFILE.**




```
<packidge id="REGISTRATION_PROFILE" userprofile="profile-registration" score="$score" weight="$weight" >
   <topic>Benutzerprofil</topic>
   <name>Registrierung</name>
   <page index="1" topic="willkommen">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/registrierung/1_willkommen.mp4"/>
         <text>Herzlich willkommen bei der Anmeldung. Wir benötigen Angaben wie deinen Namen, dein Alter und andere Informationen zu deine
      </content>
   </page>
   <!-- registrierung -->
   <page index="2" topic="alter">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/registrierung/2_wie-alt.mp4"/>
         <text>Zuerst möchten wir wissen, wie alt du bist. Bitte klicke an, ob du unter oder über 18 Jahre alt bist.</text>
      </content>
      <row index="0" type="options">
         <label>Ich bin ...</label>
         <option type="exclusive" goto="exit-packidge">unter 18 Jahre</option>
         <option type="exclusive">18 Jahre oder älter</option>
      </row>
   </page>
   <page index="2_2" topic="einverständnis">
      <comment>
         <content index="0" type="multimedia" hratio="360" vratio="270"/>
      </comment>
      <content index="0" type="multimedia" hratio="360" vratio="270">
         <videoset src="$base_url/registrierung/einverst.mp4"/>
```

Listing 83: REGISTRATION_PROFILE template

The **REGISTRATION_PROFILE** is the registration packidge for self-registering standard Feedback users that come from the web frontend.
One can also find static templates i.e. RENAME for users from the project or employee context that don't have to register themselves manually in Feedback.
The user category template is a small snipped for the xml configuration file for personal packages (cf. chapter 5.3).



Listing 84: User category

14.5 feedback_configurations DB-Table

This database item glues all the components together on the iLex side. Being a configurations table this relation is the central point where components are coming together. Pages become combined with bundles under consideration of the specific classes that are responsible for a validity check against an xml schema. Furthermore templates are associated with *feedback_proto_bundles* (Type 1, Type 2a or Type 2b).
The result is an iLex configuration that allows the application to generate the appropriate XML for Feedback questionnaires.

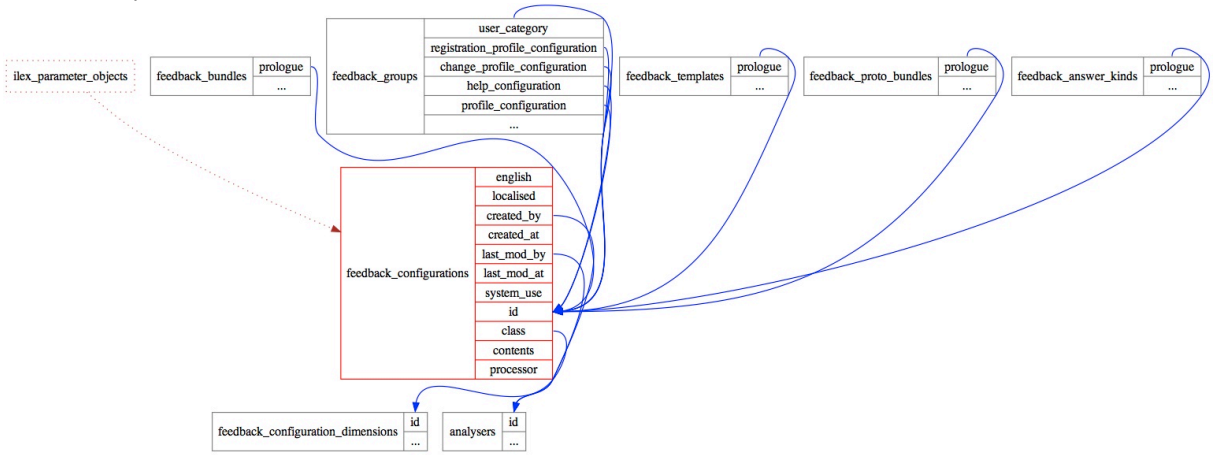


Figure 66: feedback_configurations

15. Feedback Configuration Classes

The Feedback configuration classes consider an XML schema as a class in an object oriented approach because the schema itself determines exactly what the object instance may contain.

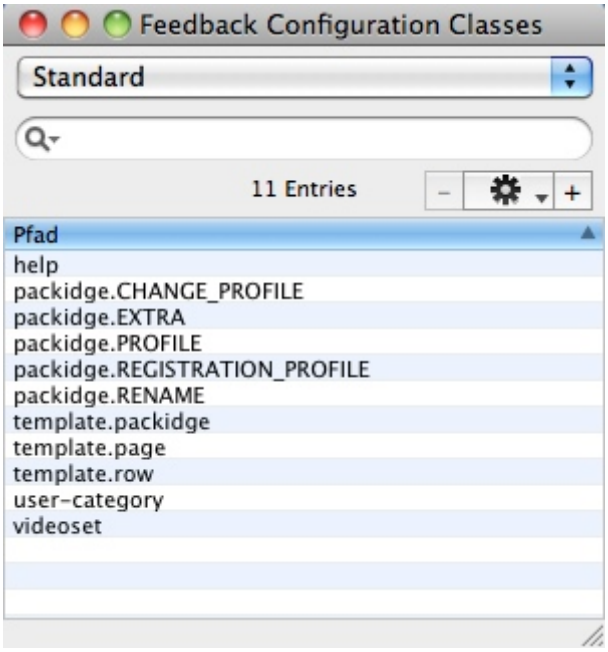


Figure 67: Feedback Configuration Classes

As we can see in the figure above there are many different types of configuration classes (feedback_configuration_dimensions) each following the basic structure of the according database table.

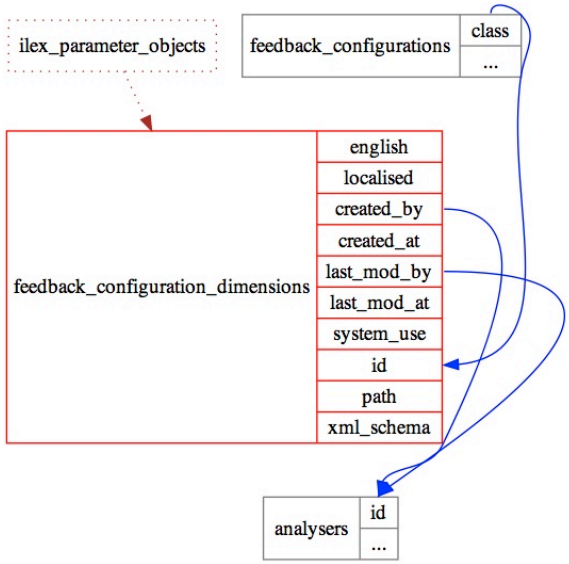


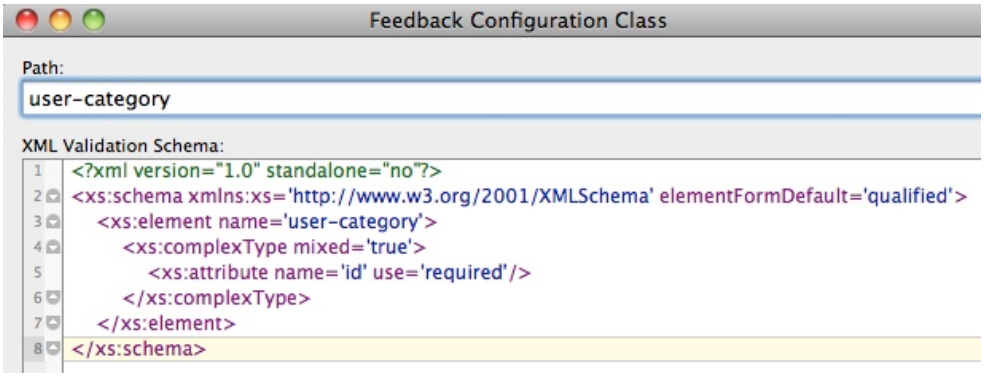
Figure 68: feedback_configuration_dimensions

By considering the table we notice the xml_schema reference as a column. This points to a xml schema implementation such as for the user category template. In order to keep the example as simple as possible this listing seems to be appropriate.

```
<?xml version="1.0" standalone="no"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' elementFormDefault='qualified'>
    <xs:element name='user-category'>
        <xs:complexType mixed='true'>
            <xs:attribute name='id' use='required'/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Listing 85: XML-Schema

Please note – the schema validation process checks for the occurrence of the mandatory attribute "id" inside the "user-category" xml element.
The next figure shows the representation of the editor functionality in the iLex GUI.



Listing 86: feedback configuration class for user category template

16. Feedback Proto Bundles

The feedback proto bundles are part of a feedback configuration and determine a content-related correlation to a certain questionnaire type such as "Type 1, Type 2a, Type 2b".

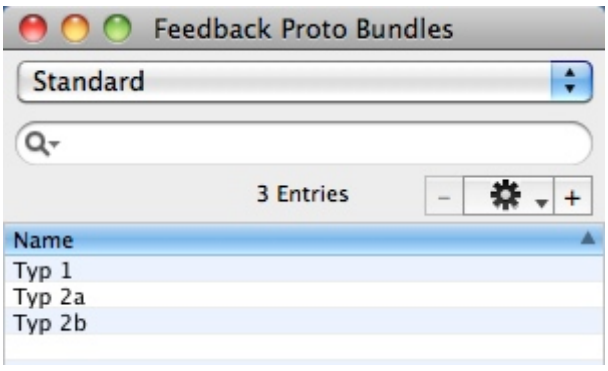


Figure 69: Feedback Proto Bundles

The above-mentioned correlation between proto bundles and feedback configurations on database level becomes illustrated as follows.

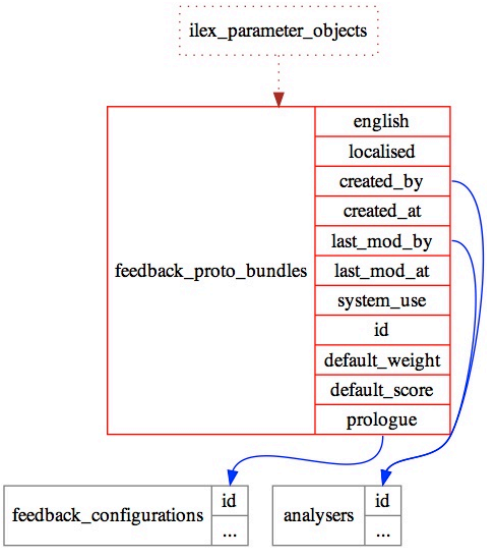


Figure 70: feedback proto bundles table

The editor screen in iLex for the proto bundles is shown by the following figure.

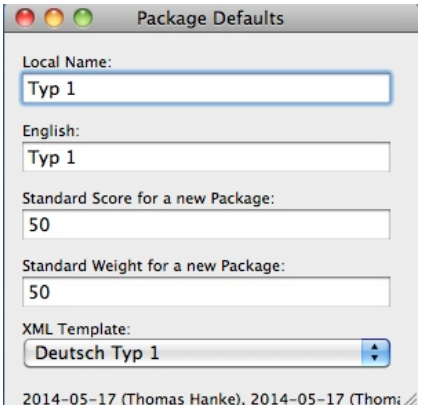


Figure 71: editor screen in iLex for the proto bundles

17. Feedback-Assets

feedback_assets for videos in packidge and template *feedback_configurations* are the places where to specify the video snippet to be shown as well as the corresponding poster (cmp. Figure 73).
N.B.: If you reference videosets in the *feedback_configurations* xml data, new *feedback_assets* records are not created automatically. Use the gear menu in the Feedback Assets window to have iLex create extra assets to match all references. Likewise, assets not (no longer) needed are not purged automatically, but they clearly become visible in the Feedback Assets list as having uses=0.
Feedback Assets are the transcribed video ressources that are generated from a HD source file for web requirements (file size, streaming appropriate). As the next figure shows we are able to look up in which template packages which videos become used ("Uses" tab).

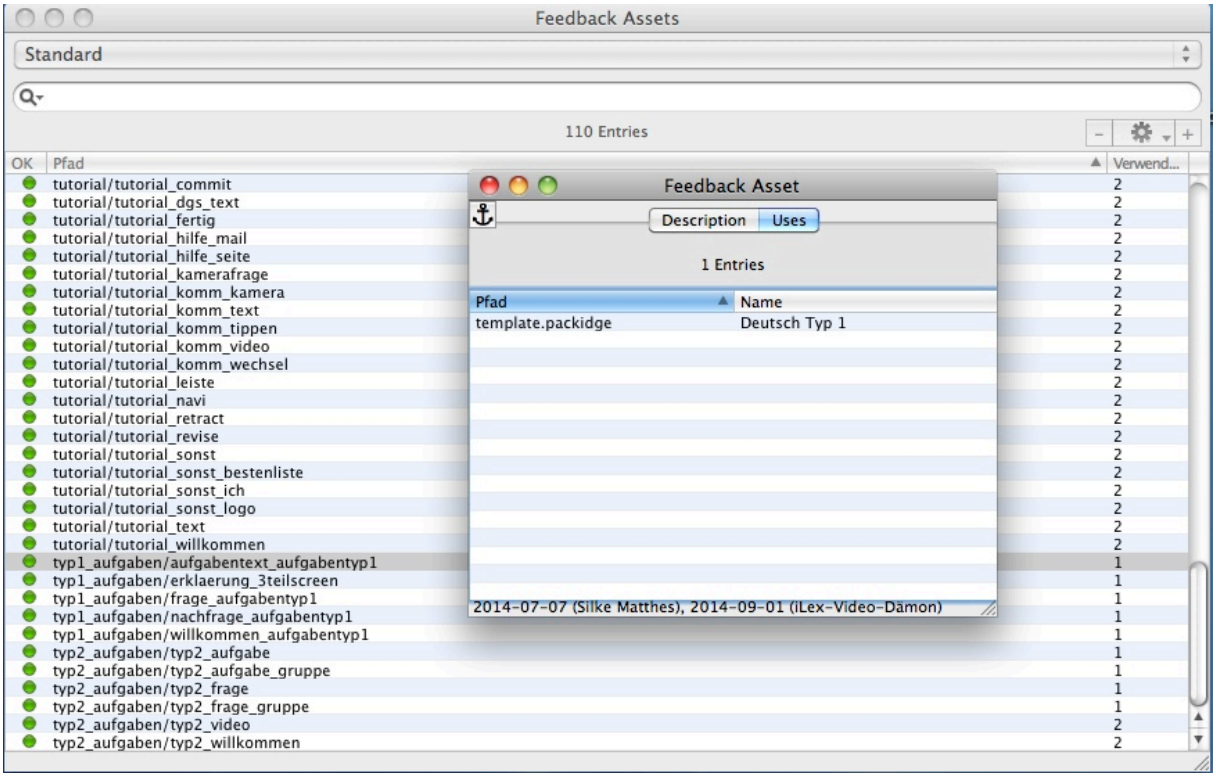


Figure 72: Feedback Assets

The integration of the assets into the general application context shows the database relations below.

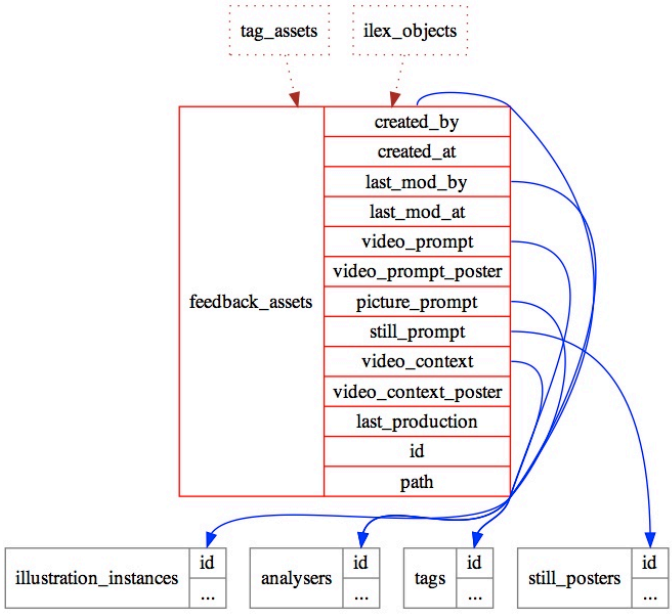


Figure 73: feedback_assets table

An asset becomes automatically created if a video entry has been added to the xml template. A newly created asset can also be linked to the template by dragging. A frame for the still has to be assigned as well.
Feedback movies become newly produced if they aren't already present on the server. If Feedback movies are supposed to be replaced, the previous version has to be deleted.

18. Further database tables for parameters

Besides the already considered *feedback_proto_bundles*, *feedback_configurations* and *feedback_configuration_dimensions* there are some further important parameter tables in context of the iLex database.
These tables are

- feedback_templates
- feedback_answer_kinds

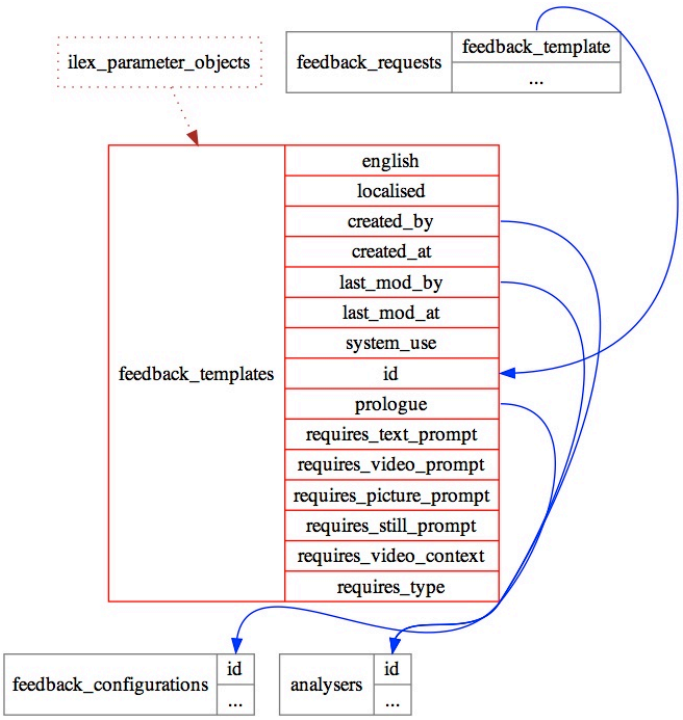


Figure 74: feedback templates table

As we can see here the templates itself have an equivalent inside the database, too. Please note the relationship between a *feedback_request* and a *feedback_template* (N:1 – Figure 53).
The *feedback_answer_kinds* are illustrated as follows.

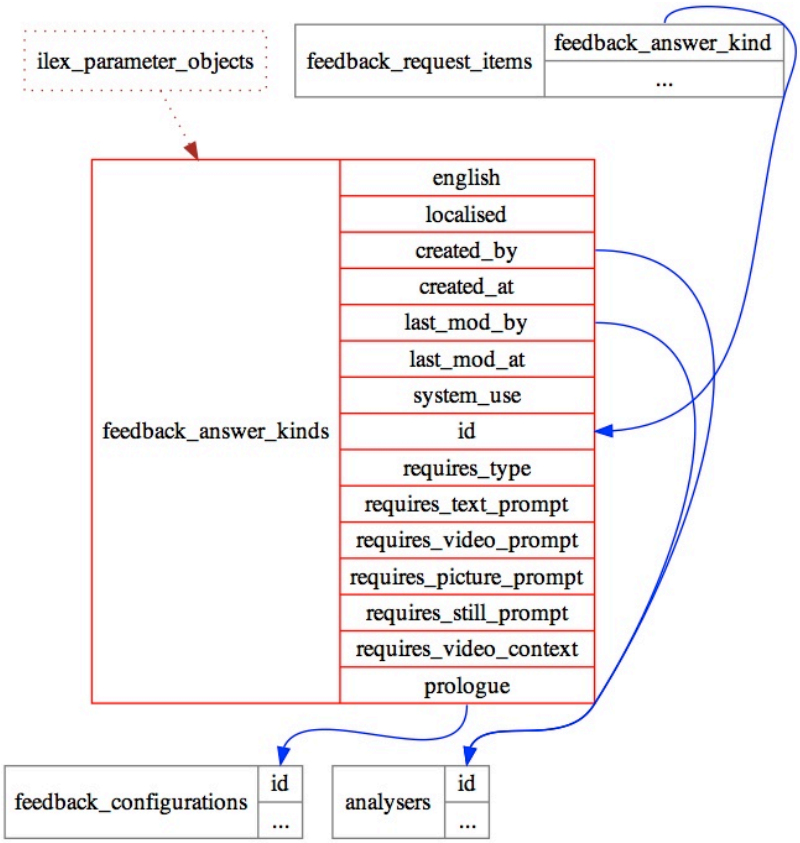


Figure 75: feedback_answer_kinds

19. iLex and Feedback users and groups

The group concept in iLex does not completely match with the roles in the feedback system. Instead, several *feedback_groups* may share a path name which corresponds to the role name in the feedback system. One of these groups sharing a path is the primary group defining the *feedback_levels* applicable for all groups with this path. Also, there must not be more than one group per path that allows self-registration. However, this need not be the primary group. Use several groups sharing the path value if you want to mix pre-registered participants with people registering themselves in order to have a shared high-score list.
The *feedback_group_memberships* records contain the user's nickname (and it is ok to change that in iLex), but not the password. Passwords are only stored in the feedback system itself with the exception of the initial passwords assigned to pre-registered participants which is derived from the *feedback_group_memberships* record id.
Levels are one element of the gamification approach taken in the feedback system. By giving users a target to reach by answering more feedback packidges, they hopefully feel encouraged to stay engaged. At the same time, the levels can be used to make sure that more difficult-to-answer packidges are not delivered to newbies by deploying same only more advanced levels. *feedback_levels* are defined for each primary group effective for all associated groups as well.
There is no need, however, to use distinct names for the levels in each primary group. Thereby you can define e.g. 4 levels appearing to all participants.

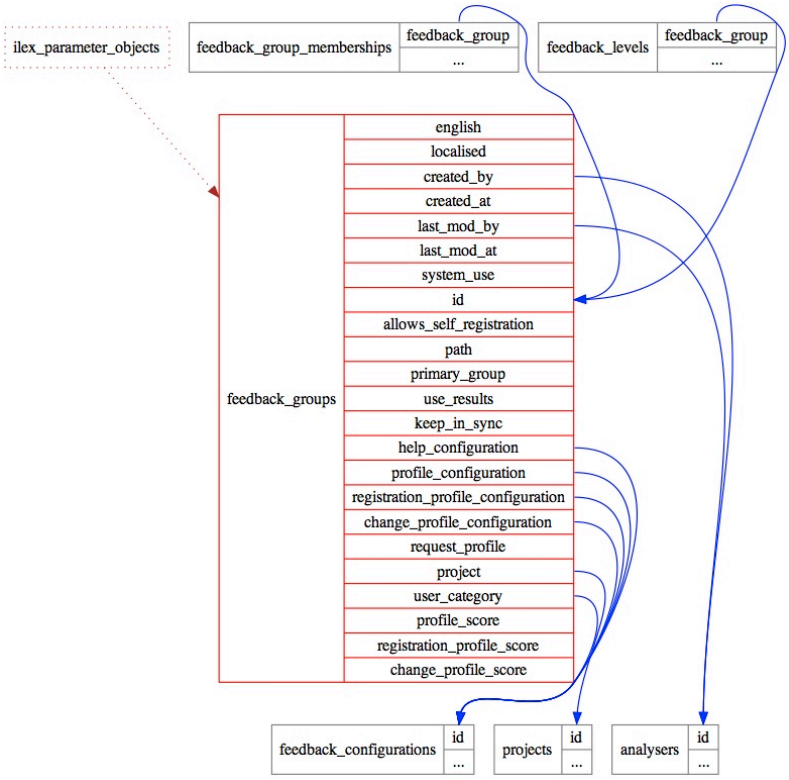


Figure 76: feedback_groups

The integration of the group concept is illustrated in the figure above. The *group_memberships* stand in relation to their groups.
The representation of *feedback_levels* can be seen below. These are meant to grant the user access to certain questionnaires when achieving a certain score limit.

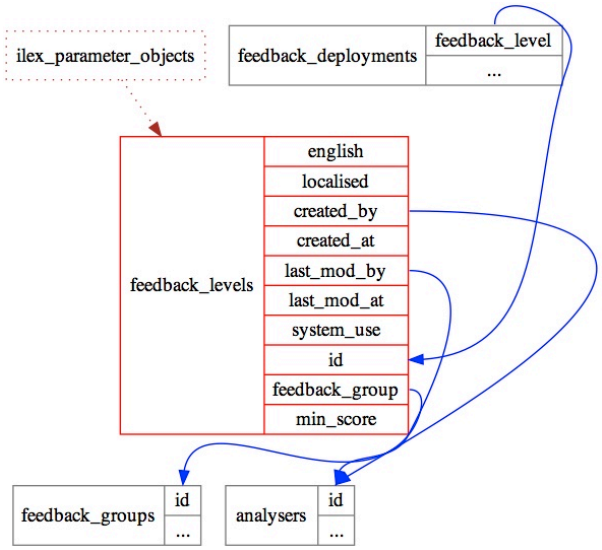


Figure 77: feedback_levels

19.1 Mapping of groups

There is a mapping mismatch between the usergroups of iLex and the Feedback groups. This becomes more obvious by regarding the standard Feedback usergroup. On the iLex side the Feedback standard usergroup for example divides into three different groups, namely

- Underage Users (Minderjährige)
- Self-Registering Users (Selbstregistrierer)
- Informants (Informanten)

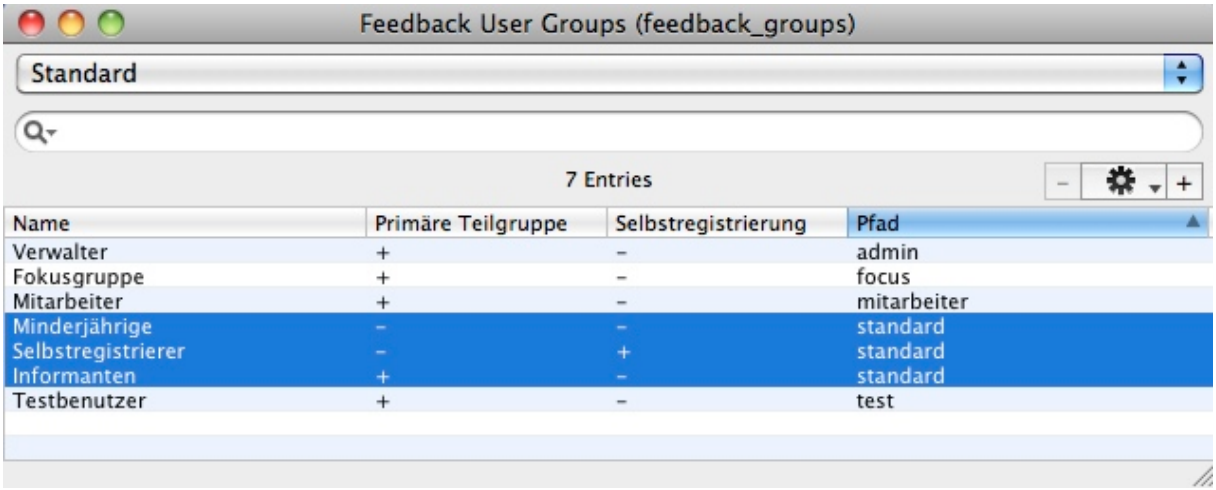


Figure 78: Feedback User Groups (Data → Parameters → Feedback User Groups)

If we want to combine the Feedback standard group with i.e. a help.xml file from the side of iLex we have to choose one out of the three corresponding iLex groups as a primary group where we

implement the association to the help.xml file. The remaining two groups therefore become secondary groups.

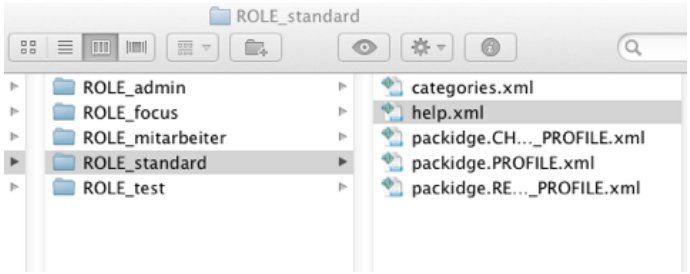


Figure 79: ROLES inside of the Feedback file system

The next figure shows the primary group “Informants” and the secondary group “Self-Registering Users”. As we can see, for both groups a standard group target for Feedback does exist in the “path” selectbox. The group on the left is marked as primary group explicitly.

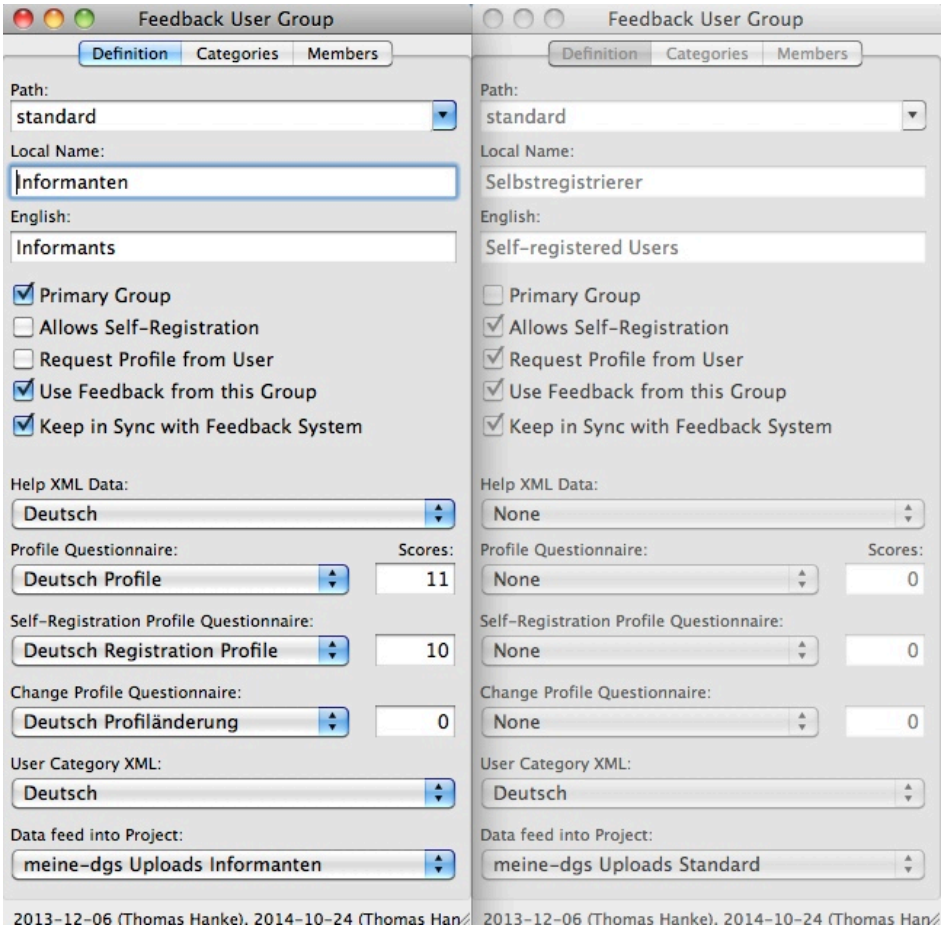


Figure 80: Primary vs. secondary group

The categorie constructs - well known from Feedback - are configured at this point as well. This is why people are able to see certain questionnaires only at a certain score level.

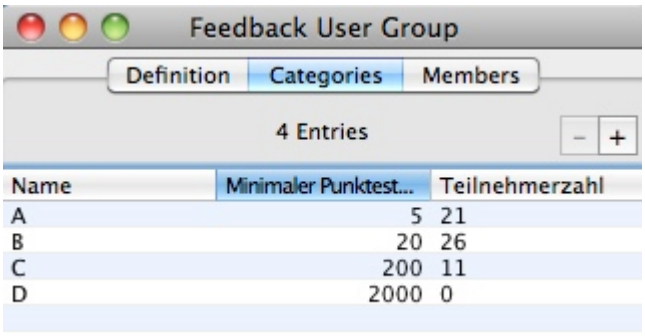


Figure 81: Feedback categories in iLex

Usergroup “Informants” and its members (excerpt):

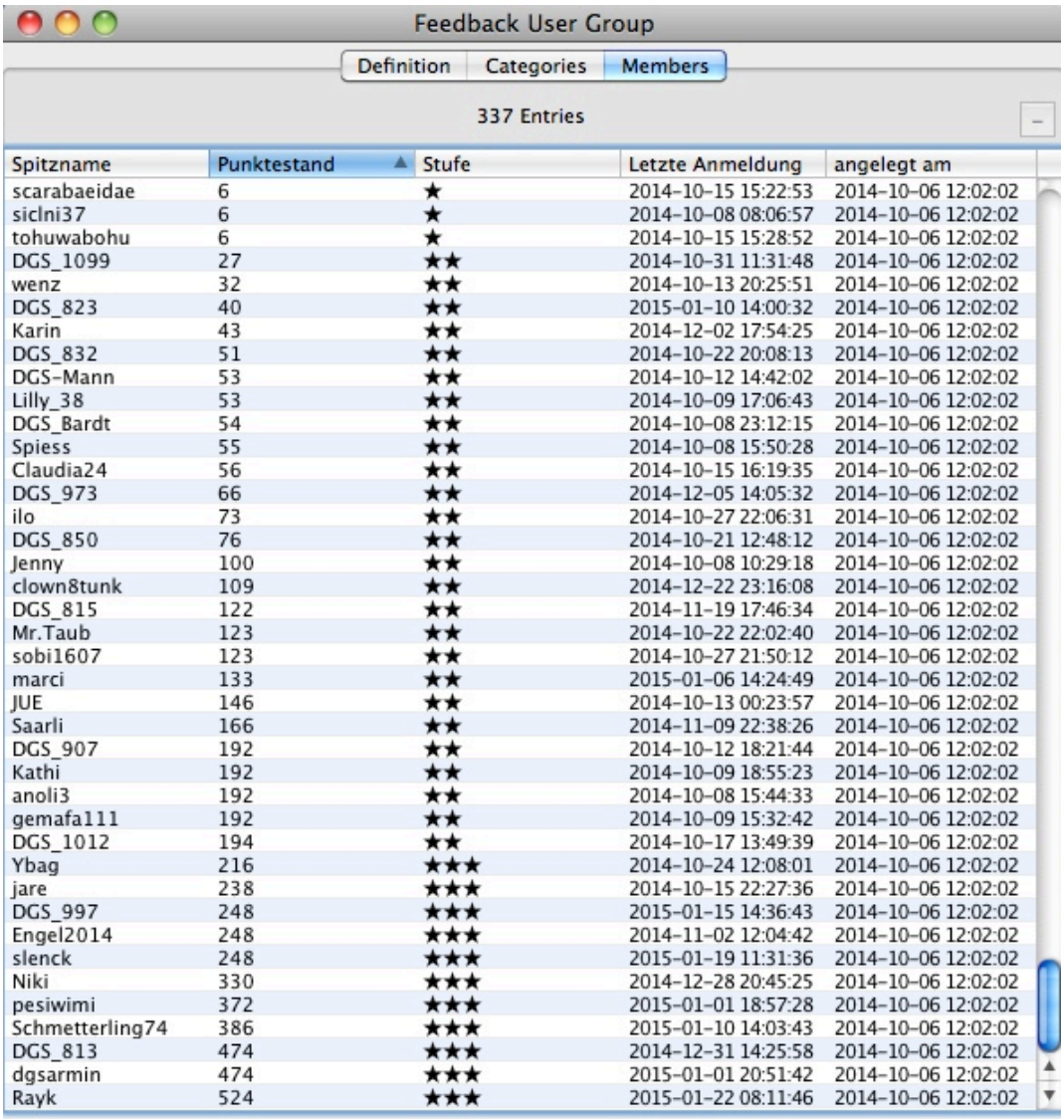


Figure 82: Usergroup “Informants” and its members (excerpt)

If we choose one single user from the list above we become aware that at this point a matching of the usrergroups has been successfully accomplished. All the iLex ressources in the infrastructure are now available for the user.

Figure 83: One single user from the list above

Summary

Let's glue all the information together including the statements from chapter 6. In the Feedback system the following usergroups (roles) are available: standard, test, admin, focus and mitarbeiter whereas in iLex usergroups can be randomly created by the staff. Therefore they have to become assigned / mapped to a certain Feedback group. As we have seen above the declaration is made under "path". Respectively one iLex group is the primary group part of a Feedback group. Here it is defined which system packidges are available to the whole Feedback group.

Creation of new members

An informant can possess multiple Feedback identities (i.e. focus, test). A person that is not (supposed to be) created as an informant (for testing) is created as follows: Informant use "Feedback-Test" (`Data --> Informants --> select Informant --> tab "Feedback" --> add`).

Select usergroups and the nickname (= username). The scores and registration date don't have to be inserted. You are able to reset the password at this point as well.

19.2 Registration Procedures

If you want to pre-register someone for the feedback system, you have to select a nickname and a group for that person.

1. Check if the person already exists in the database as an informant.
2. If not, create a new informant record and fill in the necessary details.
3. Now switch to the Feedback tab in the informant's detail window.
4. Click on "+" to create a new feedback participant.
5. Select the appropriate feedback group and type in a nickname. If you really want, you can also set a user score to some non-negative number although that's not really fair…
6. If you have assigned the participant to a group where users a not required to fill in a metadata questionnaire, you'd better fill out these metadata records yourself right now unless you already have them in the database for an earlier participation in one of your projects.

With the next deployment, the user record is created on the feedback server so that the user can log in using the nickname you have just created. The password is set to a default ("geheim" followed by the id of the newly created *feedback_group_memberships* record id) that the user can change at any time.

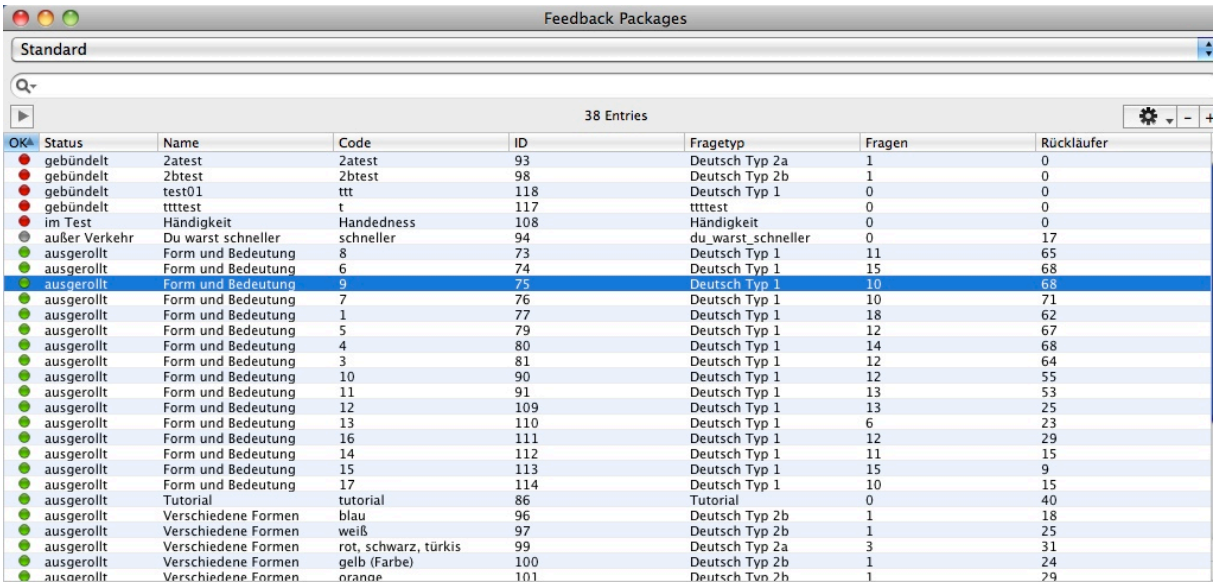
7. Let the user know about her/his nickname and password.
8. If you want to give the user the opportunity to change the nickname, create an individual deployment of a special rename questionnaire to that user.

Users registering themselves via the Feedback system should normally be asked to fill a metadata questionnaire in order to provide the data necessary for making use of their contribution, like hearing status and region they are from. Currently, self-registration is only possible for one user

group. However, it is possible to move the participants to another group once they have registered. (One possible case here is a person whom you have sent preregistered account data but who might have misplaced these information and uses self-registration instead.)

20. Representation of Feedback XML Constructs in iLex

We already know xml questionnaires from the Feedback web-application. They are created from the work inside of iLex. In the figure below we can see the deployed packages besides the packages with another status such as “in test”, “bundled” etc.



OK	Status	Name	Code	ID	Fragetyp	Fragen	Rückläufer
	gebündelt	2atest	2atest	93	Deutsch Typ 2a	1	0
	gebündelt	2btest	2btest	98	Deutsch Typ 2b	1	0
	gebündelt	test01	ttt	118	Deutsch Typ 1	0	0
	gebündelt	ttttest	t	117	ttttest	0	0
	im Test	Händigkeit	Handedness	108	Händigkeit	0	0
	außer Verkehr	Du warst schneller	schneller	94	du_warst_schneller	0	17
	ausgerollt	Form und Bedeutung	8	73	Deutsch Typ 1	11	65
	ausgerollt	Form und Bedeutung	6	74	Deutsch Typ 1	15	68
	ausgerollt	Form und Bedeutung	9	75	Deutsch Typ 1	10	68
	ausgerollt	Form und Bedeutung	7	76	Deutsch Typ 1	10	71
	ausgerollt	Form und Bedeutung	1	77	Deutsch Typ 1	18	62
	ausgerollt	Form und Bedeutung	5	79	Deutsch Typ 1	12	67
	ausgerollt	Form und Bedeutung	4	80	Deutsch Typ 1	14	68
	ausgerollt	Form und Bedeutung	3	81	Deutsch Typ 1	12	64
	ausgerollt	Form und Bedeutung	10	90	Deutsch Typ 1	12	55
	ausgerollt	Form und Bedeutung	11	91	Deutsch Typ 1	13	53
	ausgerollt	Form und Bedeutung	12	109	Deutsch Typ 1	13	25
	ausgerollt	Form und Bedeutung	13	110	Deutsch Typ 1	6	23
	ausgerollt	Form und Bedeutung	16	111	Deutsch Typ 1	12	29
	ausgerollt	Form und Bedeutung	14	112	Deutsch Typ 1	11	15
	ausgerollt	Form und Bedeutung	15	113	Deutsch Typ 1	15	9
	ausgerollt	Form und Bedeutung	17	114	Deutsch Typ 1	10	15
	ausgerollt	Tutorial	tutorial	86	Tutorial	0	40
	ausgerollt	Verschiedene Formen	blau	96	Deutsch Typ 2b	1	18
	ausgerollt	Verschiedene Formen	weiß	97	Deutsch Typ 2b	1	25
	ausgerollt	Verschiedene Formen	rot, schwarz, türkis	99	Deutsch Typ 2a	3	31
	ausgerollt	Verschiedene Formen	gelb (Farbe)	100	Deutsch Typ 2b	1	24
	ausgerollt	Verschiedene Formen	orange	101	Deutsch Typ 2b	1	29

Figure 84: Selection area and package views

Inside of iLex one is able to choose other package views/filters in order to change to a more detailed overview of the available package bundles.

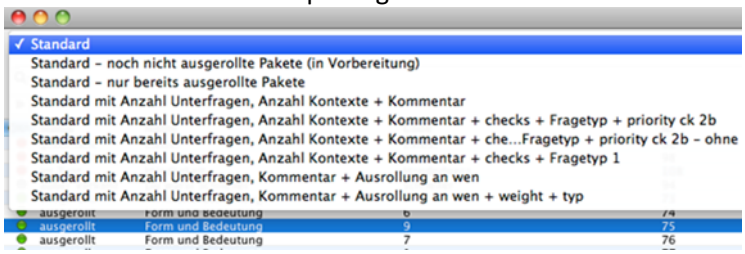
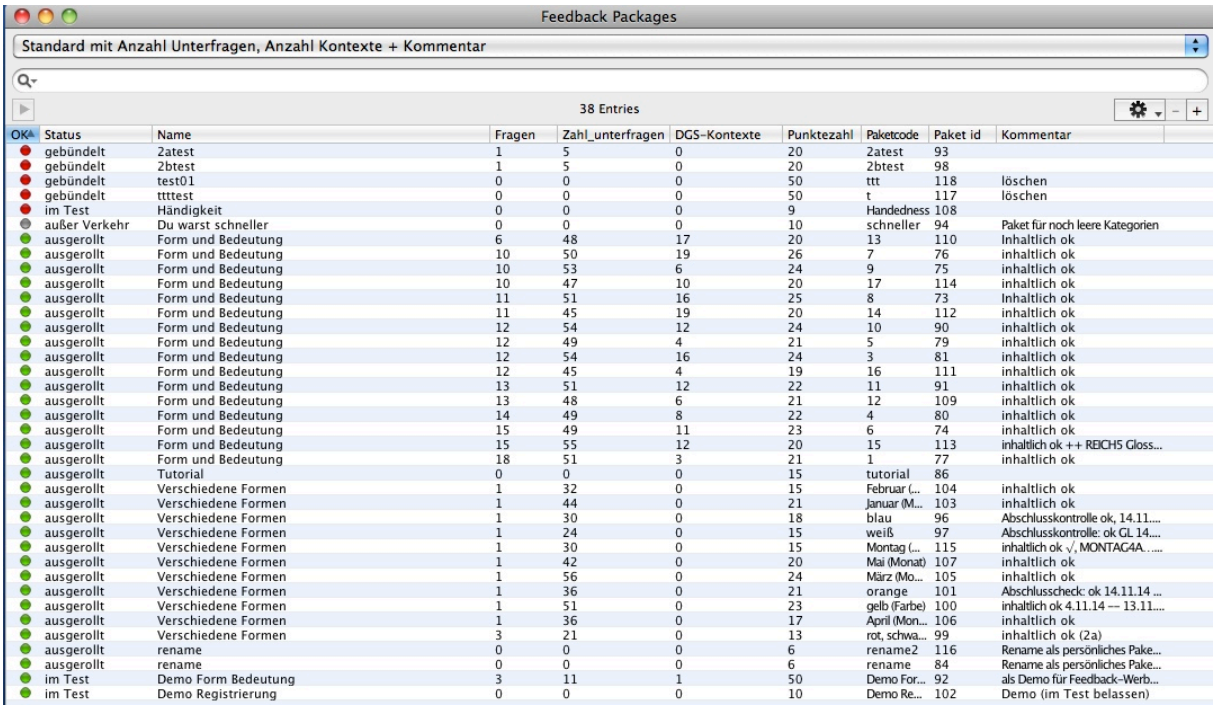


Figure 85: Filters

As can be seen in the next screen some new columns do occur inside the table view such as “score” and “comments”.



Feedback Packages

Standard mit Anzahl Unterfragen, Anzahl Kontexte + Kommentar

38 Entries

OK	Status	Name	Fragen	Zahl_unterfragen	DGS-Kontexte	Punktezahl	Paketcode	Paket id	Kommentar
	gebündelt	2atest	1	5	0	20	2atest	93	
	gebündelt	2btest	1	5	0	20	2btest	98	
	gebündelt	test01	0	0	0	50	ttt	118	löschen
	gebündelt	ttttest	0	0	0	50	t	117	löschen
	im Test	Händigkeit	0	0	0	9	Handedness	108	
	außer Verkehr	Du warst schneller	0	0	0	10	schneller	94	Paket für noch leere Kategorien
	ausgerollt	Form und Bedeutung	6	48	17	20	13	110	Inhaltlich ok
	ausgerollt	Form und Bedeutung	10	50	19	26	7	76	inhaltlich ok
	ausgerollt	Form und Bedeutung	10	53	6	24	9	75	inhaltlich ok
	ausgerollt	Form und Bedeutung	10	47	10	20	17	114	inhaltlich ok
	ausgerollt	Form und Bedeutung	11	51	16	25	8	73	Inhaltlich ok
	ausgerollt	Form und Bedeutung	11	45	19	20	14	112	inhaltlich ok
	ausgerollt	Form und Bedeutung	12	54	12	24	10	90	inhaltlich ok
	ausgerollt	Form und Bedeutung	12	49	4	21	5	79	inhaltlich ok
	ausgerollt	Form und Bedeutung	12	54	16	24	3	81	inhaltlich ok
	ausgerollt	Form und Bedeutung	12	45	4	19	16	111	inhaltlich ok
	ausgerollt	Form und Bedeutung	13	51	12	22	11	91	inhaltlich ok
	ausgerollt	Form und Bedeutung	13	48	6	21	12	109	inhaltlich ok
	ausgerollt	Form und Bedeutung	14	49	8	22	4	80	inhaltlich ok
	ausgerollt	Form und Bedeutung	15	49	11	23	6	74	inhaltlich ok
	ausgerollt	Form und Bedeutung	15	55	12	20	15	113	inhaltlich ok ++ REICH5 Gloss...
	ausgerollt	Form und Bedeutung	18	51	3	21	1	77	inhaltlich ok
	ausgerollt	Tutorial	0	0	0	15	tutorial	86	
	ausgerollt	Verschiedene Formen	1	32	0	15	Februar (...	104	inhaltlich ok
	ausgerollt	Verschiedene Formen	1	44	0	21	Januar (M...	103	inhaltlich ok
	ausgerollt	Verschiedene Formen	1	30	0	18	blau	96	Abschlusskontrolle ok, 14.11....
	ausgerollt	Verschiedene Formen	1	24	0	15	weiß	97	Abschlusskontrolle: ok GL 14....
	ausgerollt	Verschiedene Formen	1	30	0	15	Montag (...	115	inhaltlich ok √, MONTAG4A.....
	ausgerollt	Verschiedene Formen	1	42	0	20	Mai (Monat)	107	inhaltlich ok
	ausgerollt	Verschiedene Formen	1	56	0	24	März (Mo...	105	inhaltlich ok
	ausgerollt	Verschiedene Formen	1	36	0	21	orange	101	Abschlusscheck: ok 14.11.14 ...
	ausgerollt	Verschiedene Formen	1	51	0	23	gelb (Farbe)	100	inhaltlich ok 4.11.14 –– 13.11....
	ausgerollt	Verschiedene Formen	1	36	0	17	April (Mon...	106	inhaltlich ok
	ausgerollt	Verschiedene Formen	3	21	0	13	rot, schwa...	99	inhaltlich ok (2a)
	ausgerollt	rename	0	0	0	6	rename2	116	Rename als persönliches Pake...
	ausgerollt	rename	0	0	0	6	rename	84	Rename als persönliches Pake...
	im Test	Demo Form Bedeutung	3	11	1	50	Demo For...	92	als Demo für Feedback-Werb...
	im Test	Demo Registrierung	0	0	0	10	Demo Re...	102	Demo (im Test belassen)

Figure 86: New columns inside the table view

The actions that can be taken are located under the cogwheel button.

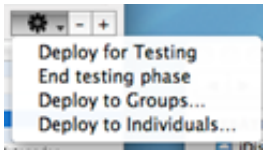


Figure 87: Deployment window

The following actions can be executed on feedback package items that are bundles:

- Deployment for testing
- Stop test stage
- Deployment to groups
- Deployment to individuals

From the side of iLex an auto-deployment is executed every 30 minutes. It can be switched to manual mode as well. Movies become only newly produced if they aren't already present. In order to force a newly production the previous movie has to be deleted.
Packages are only produced in case they are marked as "deployed" in iLex.
The window in figure 86 is located unter `Data --> Feedback Packages` and offers the aforementioned services under the cogwheel section.

- Deploy for Testing: The package is deployed being selectable via the admin account
- Deploy to Groups (Stop the testing phase before): Category has to be selected (scores and weight usually by defaults)

Please note: Type 1 packages are deployed in Category B - Category A for tutorial only.
iLex groups are described in chapter 19.

- Deploy to Individuals: packages are deployed in the personal category.

Retirement of a package
In order to retire a package open the Feedback package as shown, then switch to "Deployments", open the desired deployment and uncheck "active".

20.1 – Example of packidge 75

Refering to the example in chapter 8.2 the focus was on the xml syntax of Feedback packidges whereas here we want to concentrate on the representation of the corresponding xml constructs inside the iLex GUI. Before the packidge 75 was available for Feedback the xml was created and bundled here.
The data record ID 75 is opened under “Feedback Packages” in relation to the example of chapter 8.2.

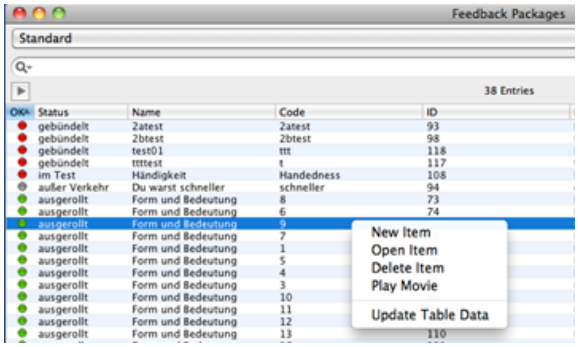


Figure 88: Data record ID 75

By opening the specific data record 75 we become able to edit its configurations such as name, code, scores, weight etc.

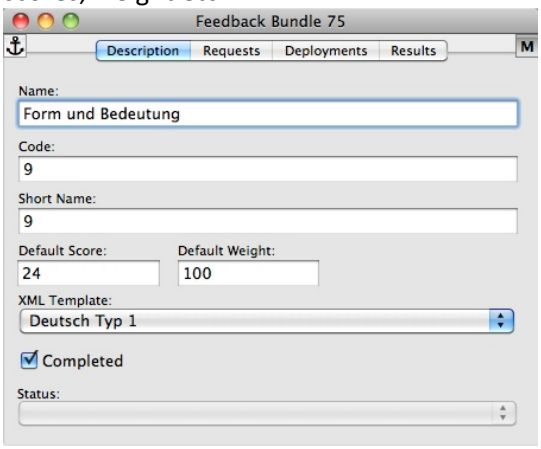


Figure 89: View of Feedback packidge 75 - Description tab

There are different possible xml configurations we can select which is nothing else than the template feedback packages discussed in chapter 14.

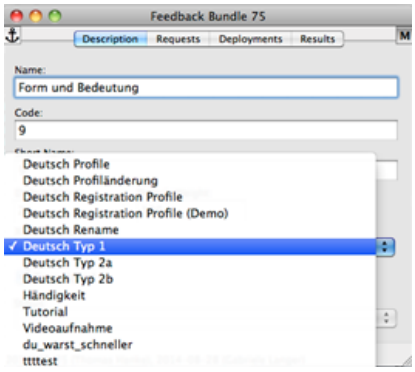


Figure 90: Possible configurations according to type 1, 2a or 2b

Now we switch to the questions of packidge 75. The iLex type „TIER4-$SAM“ corresponds to the assignment of the “topic” xml attribute of the page tag (question 123: TIER4-$SAM) inside the packidge as can be seen in the next listing.

```
<page id="1183" index="1" topic="123: TIER4-$SAM">
    <comment>
        <content type="multimedia" hratio="360" vratio="270" />
    </comment>
    <content type="multimedia" hratio="360" vratio="270">

        <image  alt="Standbild aus dem Gebärdenvideo"
        src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.jpg"
        srcset="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1_2x.jpg 2x"
    />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.m3u8" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.mp4" />
        <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/frage_aufgabentyp1.webm" />

        <text>Kennst du diese Gebärde?</text>
    </content>
</page>
```

Listing 87: Packidge 75 – topic 123

The xml entry above originates from the Feedback Request entry shown in the figure below. It is semantically integrated into the whole iLex infrastructure. This makes it possible to reference a special semantic meaning behind a sign (reading).

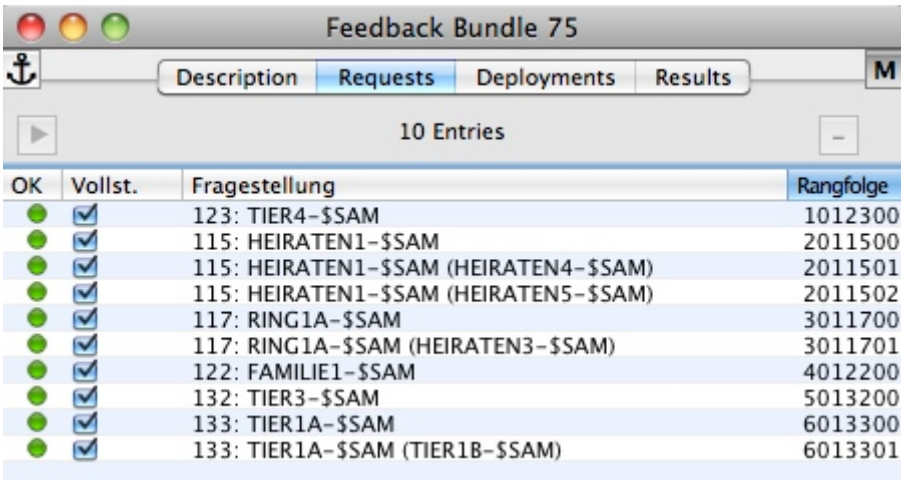


Figure 91: Packidge 75 - Requests

In order to illustrate the benefit of the integration of Feedback requests into iLex the next screenshot shows a detailed view of a Feedback request inside of iLex.

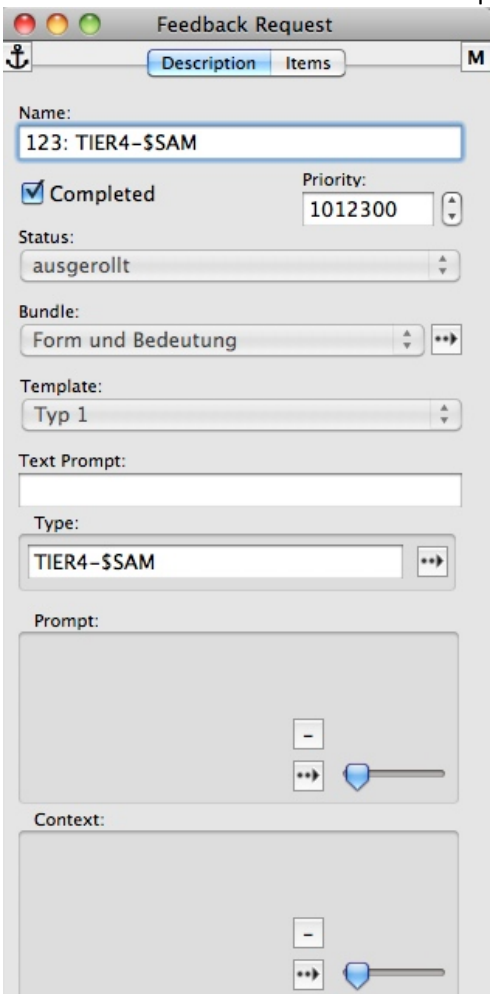


Figure 92: View of a single Feedback request

Besides the metadata we can determine a certain type for a request as well as the priority.
The next figure shows items of a Feedback request (readings) each corresponding to a row inside the XML packidge. N.B. the first record in the GUI is a static row residing inside the packidge. Therefore no request entry occurs in the list.

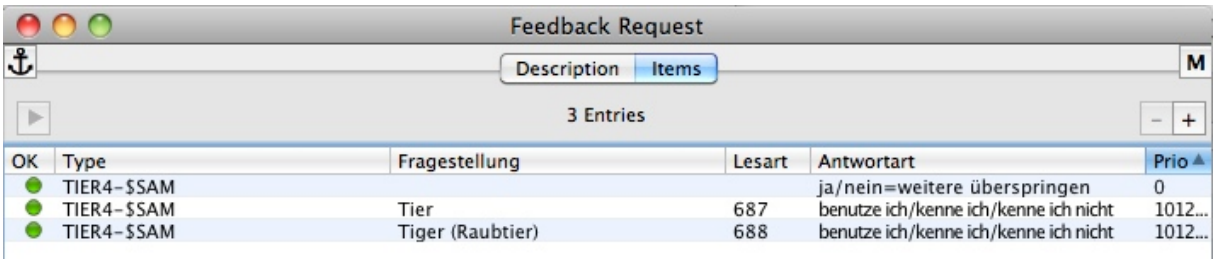


Figure 93: Elements of a Feedback request

An explicit item from the screen above can be selected and edited in the following window. For example the “answer kind” and the presented text such as „Tier“ (text prompt). Furthermore the type (here: Tier4-$SAM) can be assigned to the question.

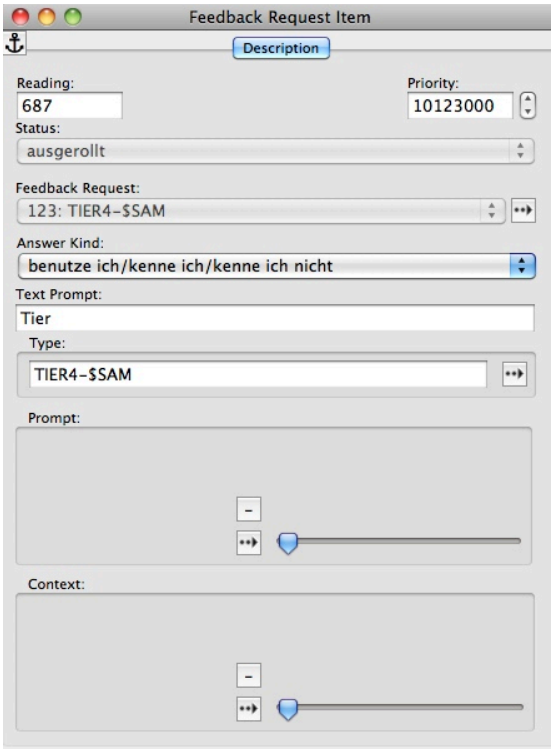


Figure 94: Details to a Feedback question / request

Depending on the desired answer button structure different button options can be selected. This selection determines the building of the row components of an xml packidge.

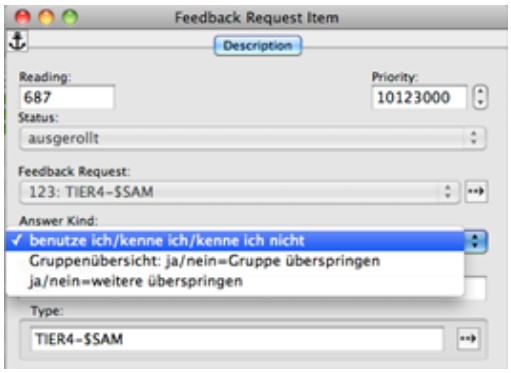


Figure 95: Answer options in a detailed Feedback question view

Under the hood we are binding here to the whole iLex infrastructure and are now able to use services such as HamNoSys notation access up to services on meanings and readings of a sign (Compare to the next screenshot).

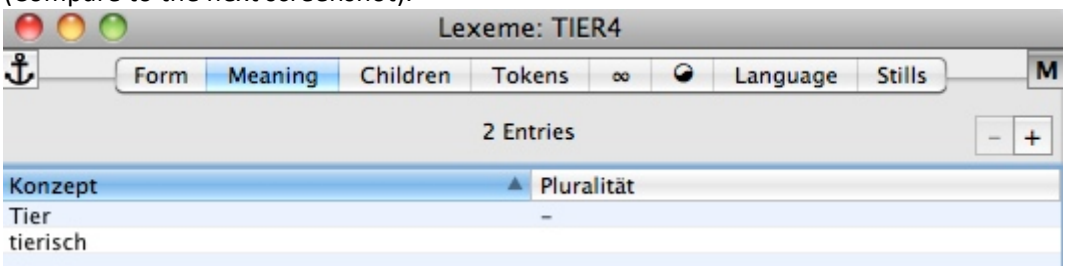


Figure 96: The meaning of a sign in form of a concept

20.2 Bundling questions as a package

We can bundle multiple requests together as a package for Feedback benefiting from the template mechanism. The staff only has to select the desired questions for a package and then choose the option “Bundle as a Package” in the upper right corner of the screen.

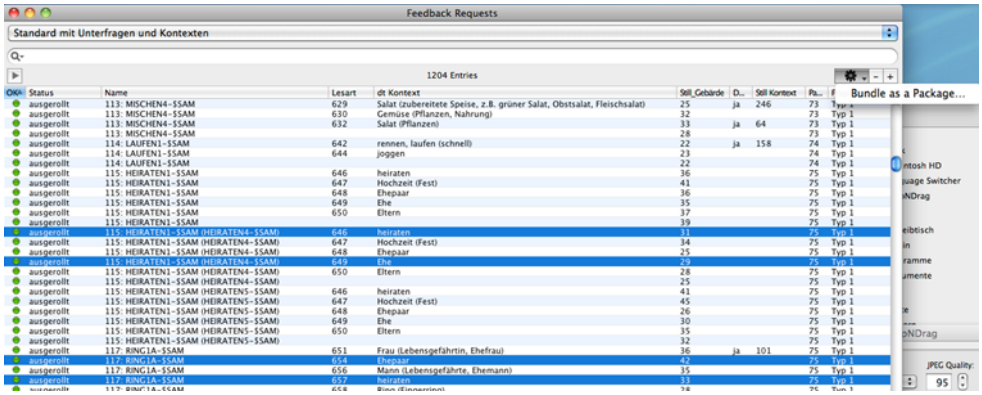


Figure 97: Bundling of a package

21. Return of questionnaires

Let us get one more step into detail by regarding the representation and access to returned/answered questionnaires in iLex. In the picture below the Feedback packidge 75 is selected again in order to illustrate the handling of returned packidges.

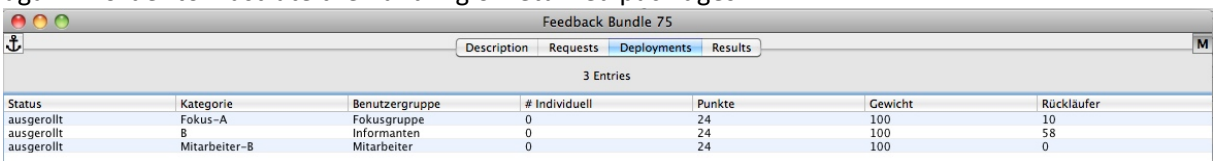


Figure 98: Feedback packidge 75 questionnaire

In the deployment overview screen there is the button “Results” which gives access to the returned questionnaires committed by the users.

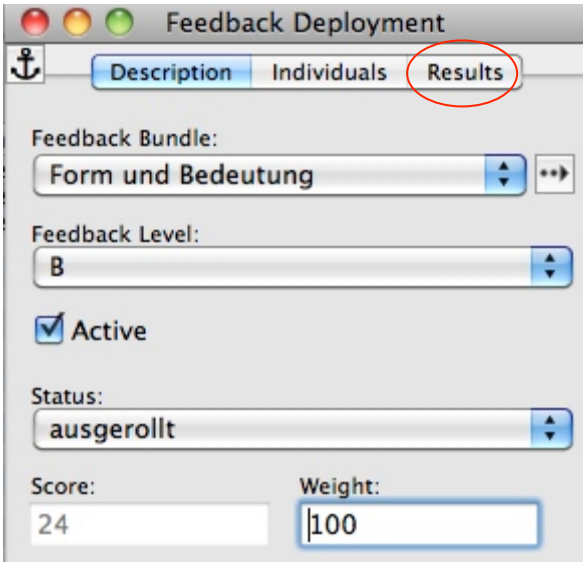


Figure 99: Results tab

We select one single returned packidge for a more detailed view.

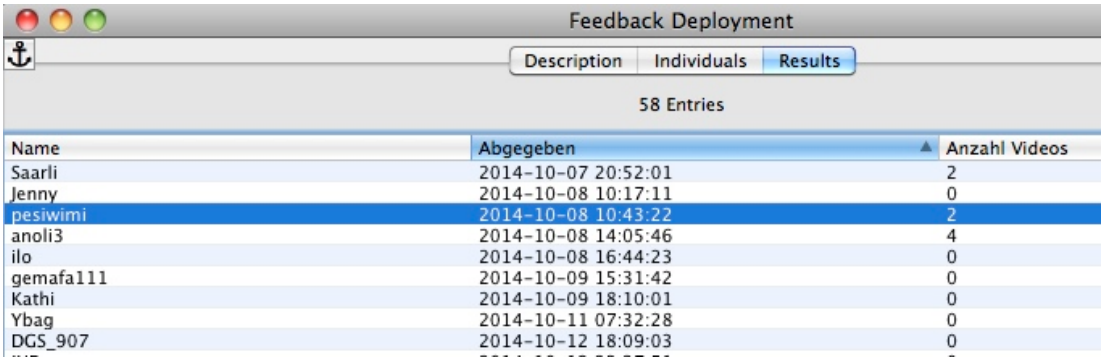
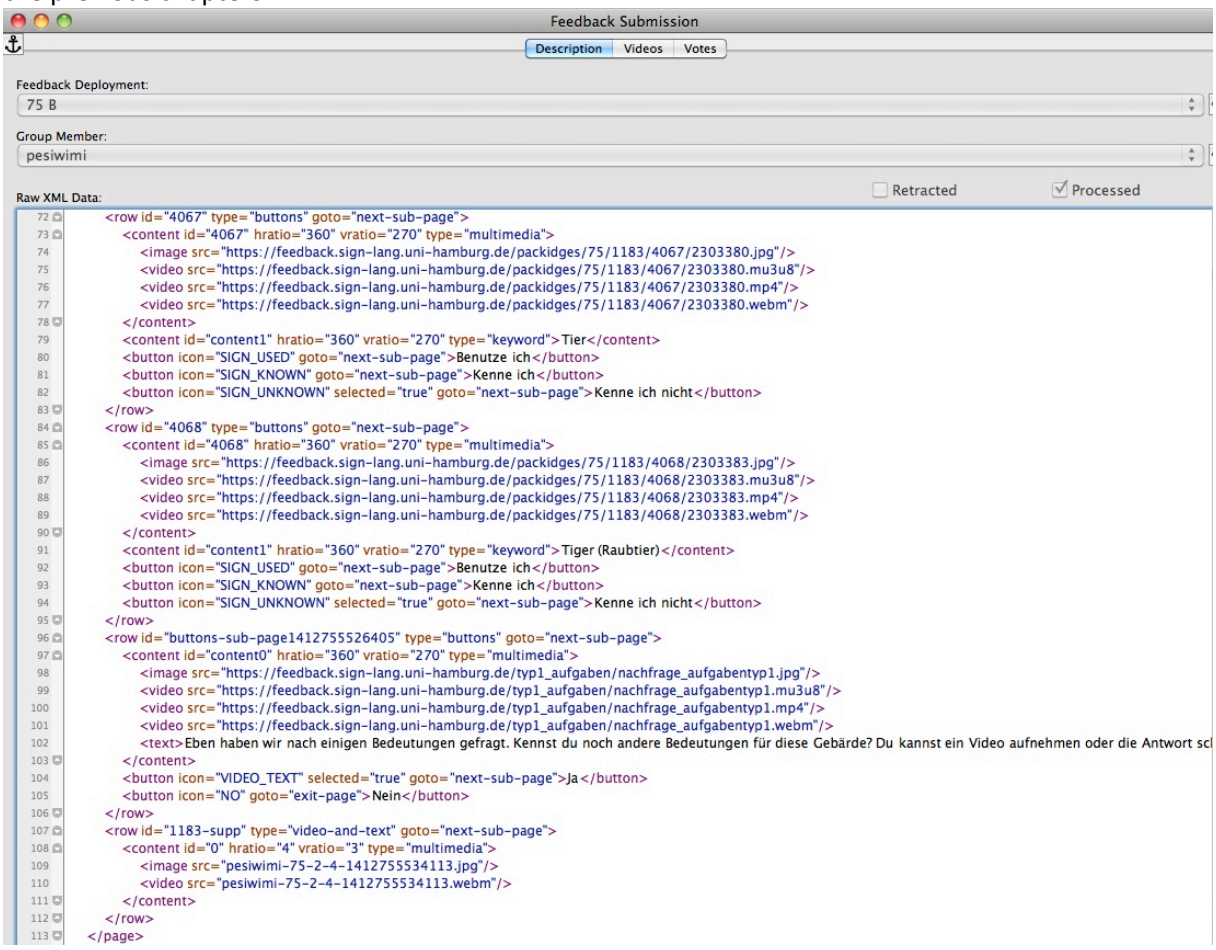


Figure 100: Feedback Deployment - Results

This leads us to the returned xml data in raw format. This is the well known xml packidge data from the previous chapters.




```
<row id="4067" type="buttons" goto="next-sub-page">
  <content id="4067" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.jpg"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.mu3u8"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.mp4"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4067/2303380.webm"/>
  </content>
  <content id="content1" hratio="360" vratio="270" type="keyword">Tier</content>
  <button icon="SIGN_USED" goto="next-sub-page">Benutze ich</button>
  <button icon="SIGN_KNOWN" goto="next-sub-page">Kenne ich</button>
  <button icon="SIGN_UNKNOWN" selected="true" goto="next-sub-page">Kenne ich nicht</button>
</row>
<row id="4068" type="buttons" goto="next-sub-page">
  <content id="4068" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4068/2303383.jpg"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4068/2303383.mu3u8"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4068/2303383.mp4"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/packidges/75/1183/4068/2303383.webm"/>
  </content>
  <content id="content1" hratio="360" vratio="270" type="keyword">Tiger (Raubtier)</content>
  <button icon="SIGN_USED" goto="next-sub-page">Benutze ich</button>
  <button icon="SIGN_KNOWN" goto="next-sub-page">Kenne ich</button>
  <button icon="SIGN_UNKNOWN" selected="true" goto="next-sub-page">Kenne ich nicht</button>
</row>
<row id="buttons-sub-page1412755526405" type="buttons" goto="next-sub-page">
  <content id="content0" hratio="360" vratio="270" type="multimedia">
    <image src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.jpg"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.mu3u8"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.mp4"/>
    <video src="https://feedback.sign-lang.uni-hamburg.de/typ1_aufgaben/nachfrage_aufgabentyp1.webm"/>
    <text>Eben haben wir nach einigen Bedeutungen gefragt. Kennst du noch andere Bedeutungen für diese Gebärde? Du kannst ein Video aufnehmen oder die Antwort sc
  </content>
  <button icon="VIDEO_TEXT" selected="true" goto="next-sub-page">Ja</button>
  <button icon="NO" goto="exit-page">Nein</button>
</row>
<row id="1183-supp" type="video-and-text" goto="next-sub-page">
  <content id="0" hratio="4" vratio="3" type="multimedia">
    <image src="pesiwimi-75-2-4-1412755534113.jpg"/>
    <video src="pesiwimi-75-2-4-1412755534113.webm"/>
  </content>
</row>
</page>
```

Listing 88: Detailed view

The xml data contains i.e. video answers from users giving more information on the meaning of a sign. In the following snippet we can see the source paths for a video answer as well as a generated preview image.

```
<row id="1183-supp" type="video-and-text" goto="next-sub-page">
  <content id="0" hratio="4" vratio="3" type="multimedia">
    <image src="pesiwimi-75-2-4-1412755534113.jpg"/>
    <video src="pesiwimi-75-2-4-1412755534113.webm"/>
  </content>
</row>
```

Listing 89: Video contribution

The contributed videos are available in the “Movies” section in the iLex GUI. This makes the handling of the data a lot easier than only relating to the xml data.

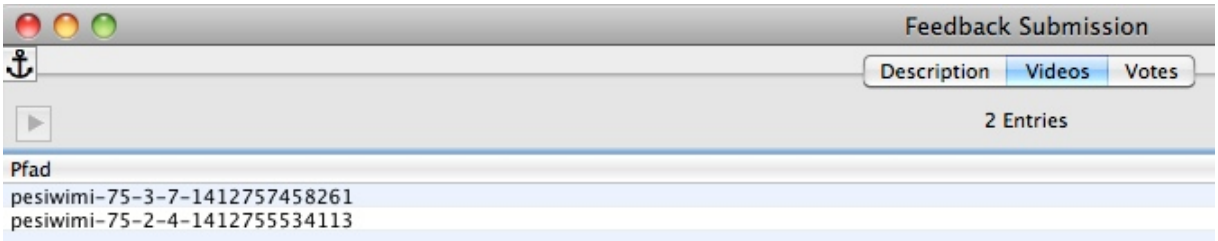
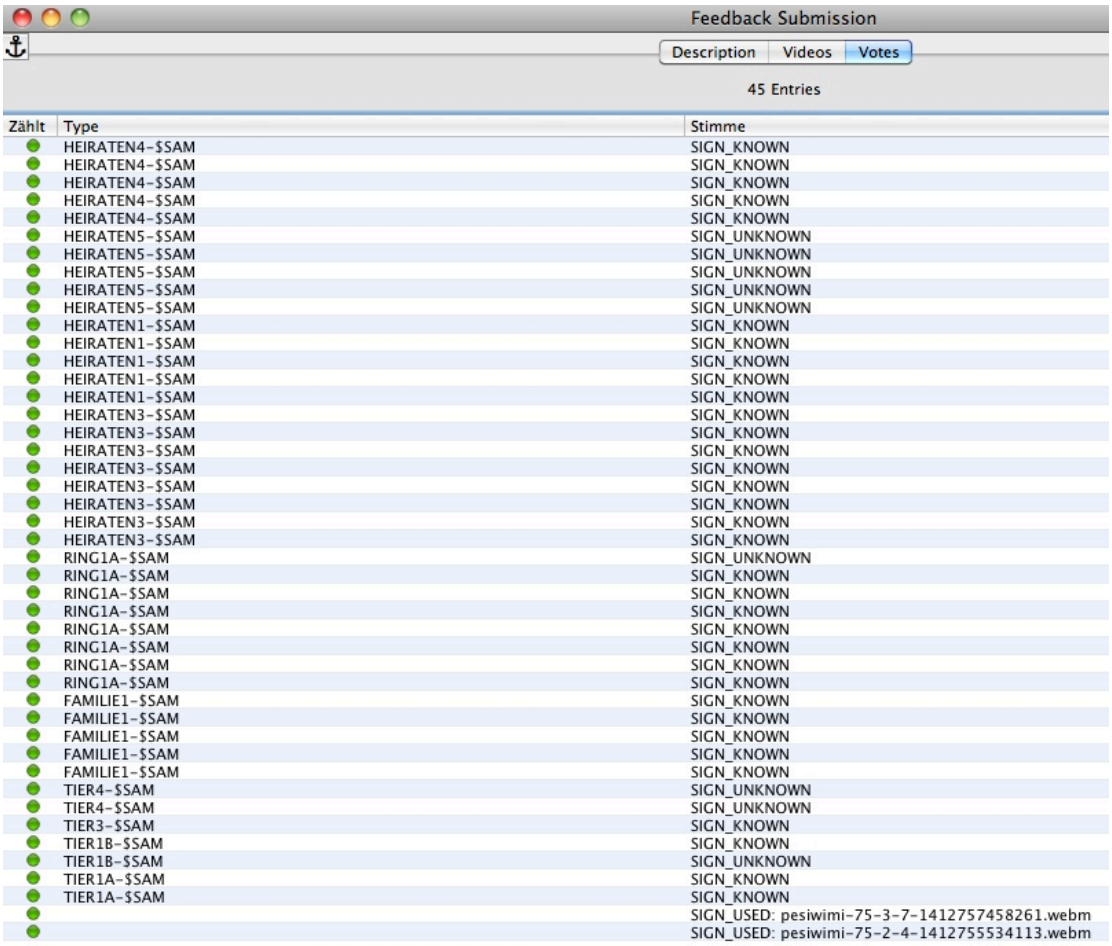


Figure 101: The video is referenced here

In the “Votes” tab we see the collected votes of a user in relation to a whole packidge. In the “Type” column the sign name is combined with the user answer including video comments as well (cf. Bottom of fig. 102).

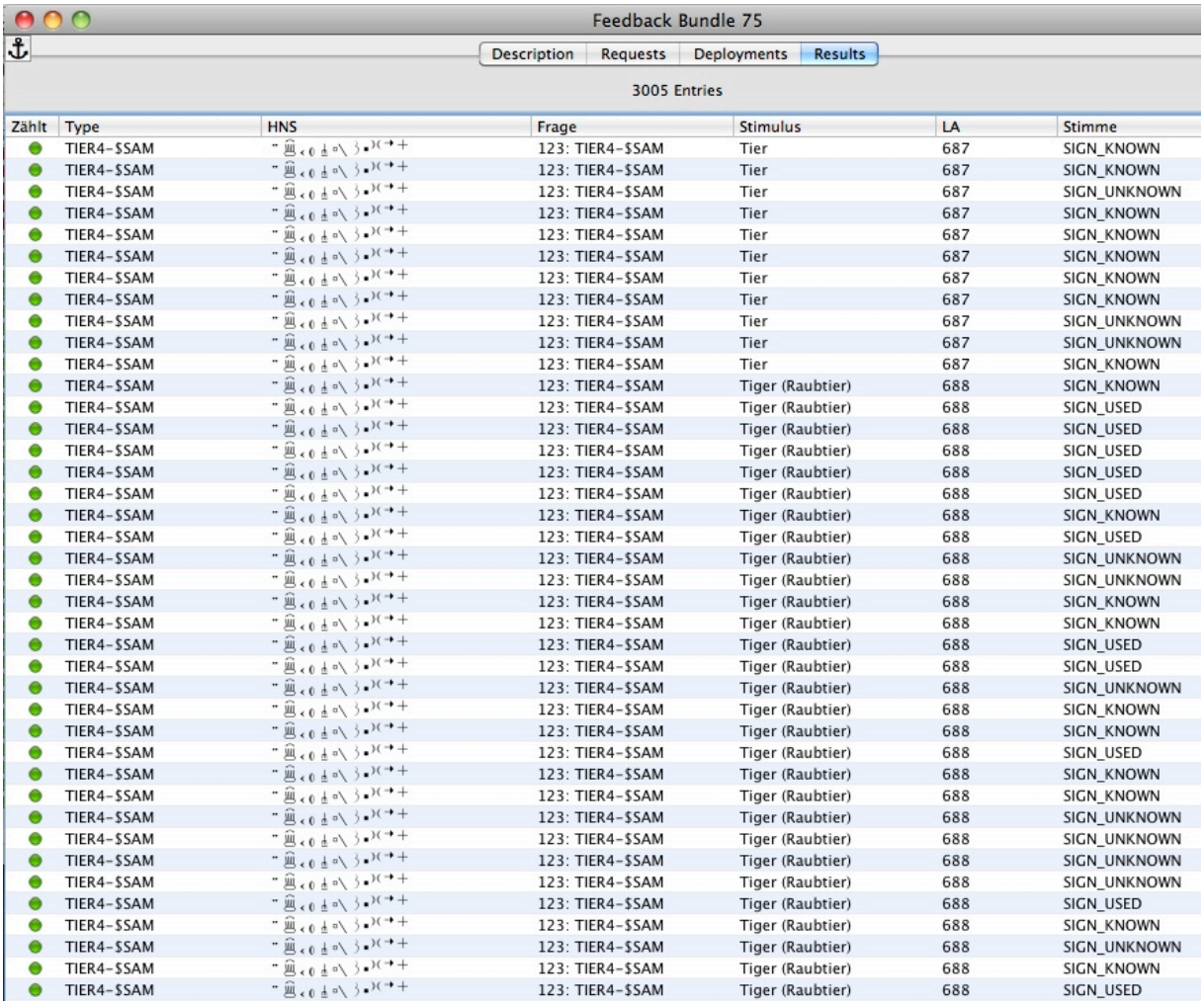


Zählt	Type	Stimme
	HEIRATEN4-$SAM	SIGN_KNOWN
	HEIRATEN4-$SAM	SIGN_KNOWN
	HEIRATEN4-$SAM	SIGN_KNOWN
	HEIRATEN4-$SAM	SIGN_KNOWN
	HEIRATEN4-$SAM	SIGN_KNOWN
	HEIRATEN5-$SAM	SIGN_UNKNOWN
	HEIRATEN5-$SAM	SIGN_UNKNOWN
	HEIRATEN5-$SAM	SIGN_UNKNOWN
	HEIRATEN5-$SAM	SIGN_UNKNOWN
	HEIRATEN5-$SAM	SIGN_UNKNOWN
	HEIRATEN1-$SAM	SIGN_KNOWN
	HEIRATEN1-$SAM	SIGN_KNOWN
	HEIRATEN1-$SAM	SIGN_KNOWN
	HEIRATEN1-$SAM	SIGN_KNOWN
	HEIRATEN1-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	HEIRATEN3-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_UNKNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	RING1A-$SAM	SIGN_KNOWN
	FAMILIE1-$SAM	SIGN_KNOWN
	FAMILIE1-$SAM	SIGN_KNOWN
	FAMILIE1-$SAM	SIGN_KNOWN
	FAMILIE1-$SAM	SIGN_KNOWN
	FAMILIE1-$SAM	SIGN_KNOWN
	TIER4-$SAM	SIGN_UNKNOWN
	TIER4-$SAM	SIGN_UNKNOWN
	TIER3-$SAM	SIGN_KNOWN
	TIER1B-$SAM	SIGN_KNOWN
	TIER1B-$SAM	SIGN_UNKNOWN
	TIER1A-$SAM	SIGN_KNOWN
	TIER1A-$SAM	SIGN_KNOWN
		SIGN_USED: pesiwimi-75-3-7-1412757458261.webm
		SIGN_USED: pesiwimi-75-2-4-1412755534113.webm

Figure 102: The votes incuding the videos im webm

22. Representation of results from a type point-of-view

Referring to the example of packidge 75 again the user answers are considered here from an iLex point-of-view. As you can see in the following screen the iLex list of Feedback Bundle 75 is sorted by types. The types are aggregated in the “Results” tab. Here the type “TIER4-$SAM” is in focus again. In the next window this type becomes related to all the votes that have been sent back to iLex from all users.



Zählt	Type	HNS	Frage	Stimulus	LA	Stimme
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tier	687	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_UNKNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_KNOWN
	TIER4-$SAM		123: TIER4-$SAM	Tiger (Raubtier)	688	SIGN_USED

Figure 103: Packidge 75 results overview

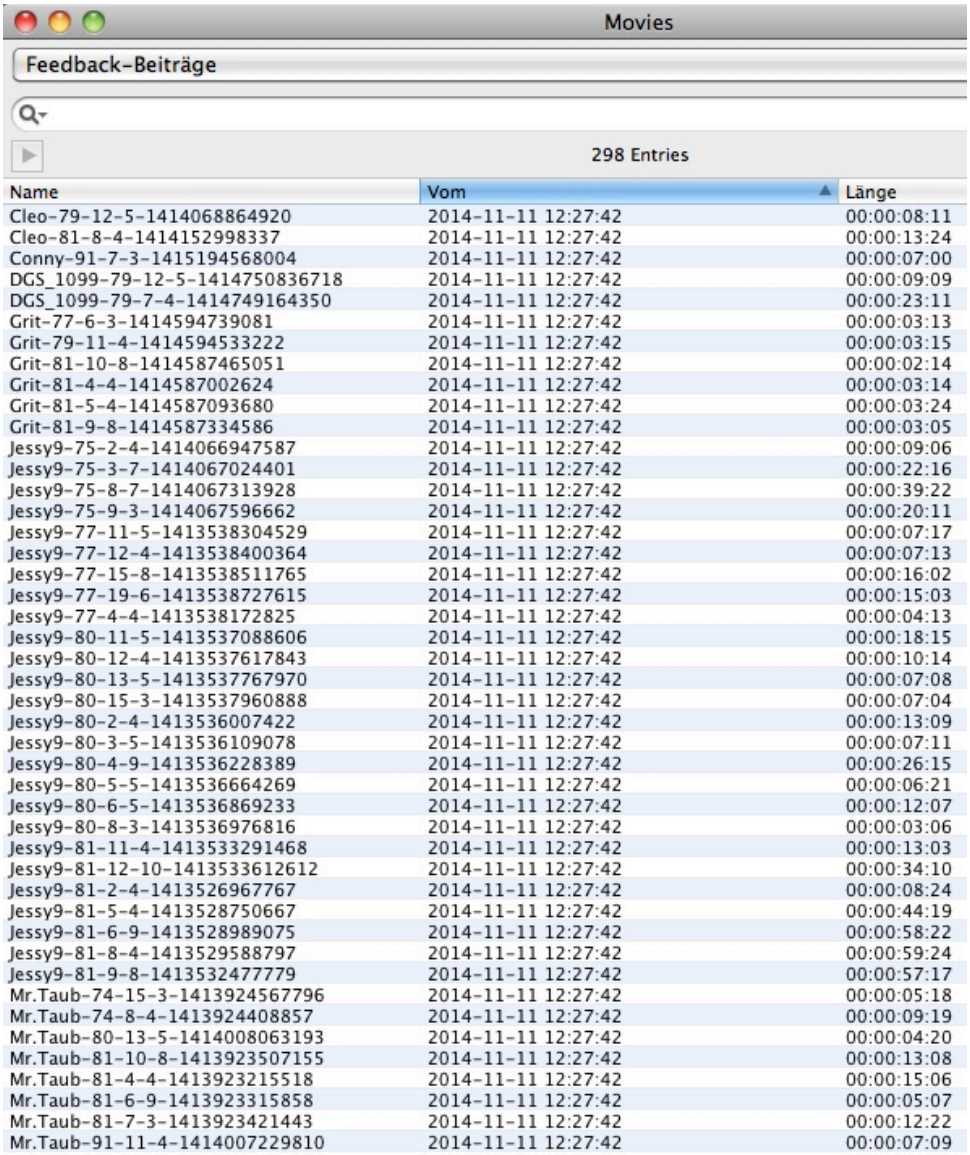
The result column does also contain the user comments to a current type.

SIGN_USED: VERLOBEN
SIGN_USED: In einigen Filmen ist nicht klar was ich bewerten soll. z.B. Familie (mit F-Handform) und dann mit Gebärdensatz ...
SIGN_USED: anoli3-75-8-7-1412769514558.webm
COMMENT-TEXT: Die Familie gebärde ich mit 2-F-Handformen und Verwandten tue ich es mit 2-V-Handformen.
SIGN_USED: Familie (z.B.: Barbe gehört zur Familie Karpfen)
COMMENT-TEXT: Kannte ich noch nicht, finde das aber interessant und lerne das jetzt neu! ;-))
SIGN_USED: Jessy9-75-8-7-1414067313928.webm
COMMENT-VIDEO: Saarli-75-Comment-1412706537461.webm
SIGN_USED: Verein (selten gesehen)

Figure 104: user comments to a current type

If you like to find out i.e. who's the author of a special comment follow these steps:
1. Drag the desired row on the notepad.
2. New query: `select * from feedback_votes where id=55481*` (id available on the notepad), *all numbers have to be adapted individually of course
3. New query: `select * from feedback_submissions where id=1170*` (id available in the preceeding query result under feedback_submission)
4. New query: `select * from feedback_group_memberships where id=41` (id available in the preceeding query result under feedback_group_membership)
-> nickname becomes shown

If you select “Movies“ in the iLex menu and also choose the ”Feedback-Contributions” filter you will get a list view of all the movies that have been sent in by the users during the answering process of questionnaires in Feedback.



Name	Vom	Länge
Cleo-79-12-5-1414068864920	2014-11-11 12:27:42	00:00:08:11
Cleo-81-8-4-1414152998337	2014-11-11 12:27:42	00:00:13:24
Conny-91-7-3-1415194568004	2014-11-11 12:27:42	00:00:07:00
DGS_1099-79-12-5-1414750836718	2014-11-11 12:27:42	00:00:09:09
DGS_1099-79-7-4-1414749164350	2014-11-11 12:27:42	00:00:23:11
Grit-77-6-3-1414594739081	2014-11-11 12:27:42	00:00:03:13
Grit-79-11-4-1414594533222	2014-11-11 12:27:42	00:00:03:15
Grit-81-10-8-1414587465051	2014-11-11 12:27:42	00:00:02:14
Grit-81-4-4-1414587002624	2014-11-11 12:27:42	00:00:03:14
Grit-81-5-4-1414587093680	2014-11-11 12:27:42	00:00:03:24
Grit-81-9-8-1414587334586	2014-11-11 12:27:42	00:00:03:05
Jessy9-75-2-4-1414066947587	2014-11-11 12:27:42	00:00:09:06
Jessy9-75-3-7-1414067024401	2014-11-11 12:27:42	00:00:22:16
Jessy9-75-8-7-1414067313928	2014-11-11 12:27:42	00:00:39:22
Jessy9-75-9-3-1414067596662	2014-11-11 12:27:42	00:00:20:11
Jessy9-77-11-5-1413538304529	2014-11-11 12:27:42	00:00:07:17
Jessy9-77-12-4-1413538400364	2014-11-11 12:27:42	00:00:07:13
Jessy9-77-15-8-1413538511765	2014-11-11 12:27:42	00:00:16:02
Jessy9-77-19-6-1413538727615	2014-11-11 12:27:42	00:00:15:03
Jessy9-77-4-4-1413538172825	2014-11-11 12:27:42	00:00:04:13
Jessy9-80-11-5-1413537088606	2014-11-11 12:27:42	00:00:18:15
Jessy9-80-12-4-1413537617843	2014-11-11 12:27:42	00:00:10:14
Jessy9-80-13-5-1413537767970	2014-11-11 12:27:42	00:00:07:08
Jessy9-80-15-3-1413537960888	2014-11-11 12:27:42	00:00:07:04
Jessy9-80-2-4-1413536007422	2014-11-11 12:27:42	00:00:13:09
Jessy9-80-3-5-1413536109078	2014-11-11 12:27:42	00:00:07:11
Jessy9-80-4-9-1413536228389	2014-11-11 12:27:42	00:00:26:15
Jessy9-80-5-5-1413536664269	2014-11-11 12:27:42	00:00:06:21
Jessy9-80-6-5-1413536869233	2014-11-11 12:27:42	00:00:12:07
Jessy9-80-8-3-1413536976816	2014-11-11 12:27:42	00:00:03:06
Jessy9-81-11-4-1413533291468	2014-11-11 12:27:42	00:00:13:03
Jessy9-81-12-10-1413533612612	2014-11-11 12:27:42	00:00:34:10
Jessy9-81-2-4-1413526967767	2014-11-11 12:27:42	00:00:08:24
Jessy9-81-5-4-1413528750667	2014-11-11 12:27:42	00:00:44:19
Jessy9-81-6-9-1413528989075	2014-11-11 12:27:42	00:00:58:22
Jessy9-81-8-4-1413529588797	2014-11-11 12:27:42	00:00:59:24
Jessy9-81-9-8-1413532477779	2014-11-11 12:27:42	00:00:57:17
Mr.Taub-74-15-3-1413924567796	2014-11-11 12:27:42	00:00:05:18
Mr.Taub-74-8-4-1413924408857	2014-11-11 12:27:42	00:00:09:19
Mr.Taub-80-13-5-1414008063193	2014-11-11 12:27:42	00:00:04:20
Mr.Taub-81-10-8-1413923507155	2014-11-11 12:27:42	00:00:13:08
Mr.Taub-81-4-4-1413923215518	2014-11-11 12:27:42	00:00:15:06
Mr.Taub-81-6-9-1413923315858	2014-11-11 12:27:42	00:00:05:07
Mr.Taub-81-7-3-1413923421443	2014-11-11 12:27:42	00:00:12:22
Mr.Taub-91-11-4-1414007229810	2014-11-11 12:27:42	00:00:07:09

Figure 105: Feedback movies

This makes it possible to show detailed overviews of certain video comments. Compare the next 2 screens.

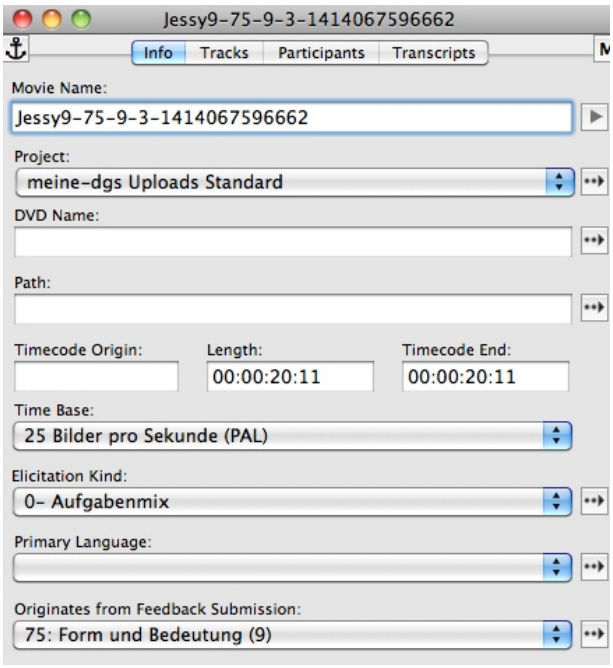


Figure 106: Info on a certain video comment that has been returned to iLex

Following this paradigm the video comment becomes entirely integrated into the iLex infrastructure and we are able to apply iLex features.

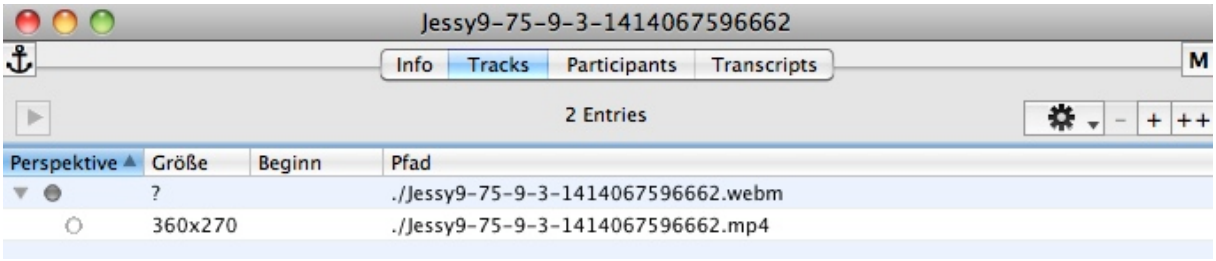


Figure 107: Tracks