

# **The ABCD EmpWeb manual, draft v1**

by Egbert de Smet

## Table of Contents

I.Introduction : background and concepts of EmpWeb.....	3
I.1 Name and history.....	3
I.2 EmpWeb as an ABCD module.....	3
I.3 EmpWeb vs. ABCD Central Circulation.....	4
I.4 EmpWeb architecture and technical aspects.....	4
I.5 EmpWeb pipe-lines.....	6
I.6 EmpWeb policies.....	9
II.Installation of EmpWeb.....	11
II.1 The EmpWeb launcher.....	11
II.1.1 Embedding into the ABCD-system.....	11
II.1.2 The Jetty-server configuration.....	12
II.1.3 Java configuration.....	12
II.1.4 The CISIS-wrapper section.....	13
II.2 Empweb installation proper.....	13
II.2.1 Common.....	14
II.2.2 db.....	14
II.2.3 dbws.....	14
II.2.4 engine.....	20
II.2.5 gui.....	22
III.Using EmpWeb.....	29
III.1 System configuration.....	29
III.1.1 Initiating EmpWeb.....	29
III.1.2 Main initialisation of the EmpWeb SQL-tables.....	32
III.1.3 Configuration of libraries.....	38
III.1.4 Administration of calendars in EmpWeb.....	41
III.1.5 Creation of new operators.....	43
III.1.6 Linking the ABCD-users and Loanobjects databases.....	46
III.1.7 Definition of policies.....	50
III.1.8 Definition of profiles by user type and object type.....	50
III.2 Using the daily transactions.....	55
III.2.1 Checking the availability of a book.....	55
III.2.2 Reservations with EmpWeb.....	56
III.2.3 Issuing a book.....	57
III.2.4 Returning a book.....	60
III.2.5 Renewing a loan-transaction.....	62
III.2.6 Reservation.....	63
III.2.7 Dealing with fines and suspensions.....	64
III.2.8 Reports and statistics.....	65
This concludes the overview of available statistics in EmpWeb, and indeed also the discussion of the use of the EmpWeb interface.....	68
IV. Annexes.....	69
.....	69
IV.1 EmpWeb Pipelines and Groovy.....	69
IV.1.1 The concept of Pipelines and Groovy.....	69
IV.1.2 Detailed discussion of a pipeline.....	70

*The ABCD EMPWEB manual*

IV.1.3 Detailed analysis of some basic classes of EmpWeb when running a transaction.....	73
IV.1.4 Structure of a process or rule in a pipeline.....	85

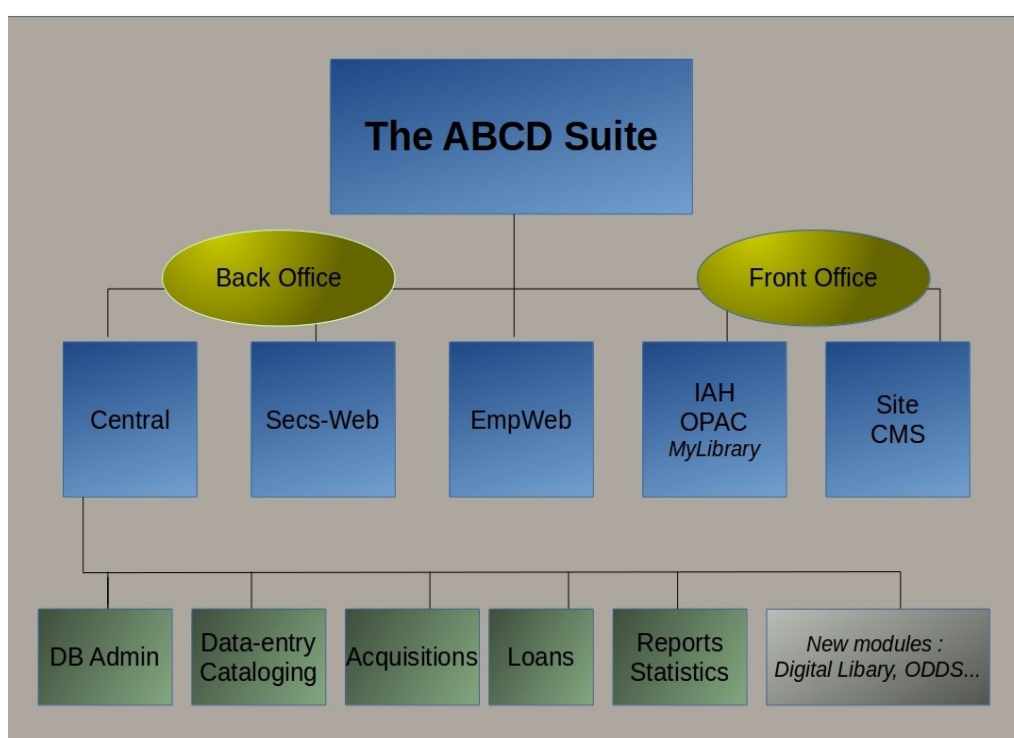
# I. Introduction : background and concepts of EmpWeb

## I.1 Name and history

EmpWeb or 'Emprestamos en Web' (Portuguese for 'loans in WWW') is an advanced Circulation module, which has been integrated into ABCD since 2010. It pre-existed as a DOS-application (Emp) and has been tested/used thoroughly in some Latin-American libraries, then transformed into a WWW-based software. As such it was acquired, from the Uruguyan company 'KALIO' which programmed both the DOS and WWW-versions, by the ABCD-team (with the support of VLIR/UOS, the Flemish (Belgian) Interuniversity Council) and fully incorporated into ABCD as a module, which could serve as an alternative to the built-in ABCD circulation sub-module in the Central module.

## I.2 EmpWeb as an ABCD module

The following overview of ABCD as a 'suite' of modules can illustrate this :



As can be seen in this scheme, EmpWeb is a 'module' of the suite at the same level of Central, Secs-Web, iAH and the Site. However, since EmpWeb does not use PHP (but Java) it will not be installed in the ABCD/www-directory, where the other modules and the ISIS-executables are found, but in its own sub-directory in the ABCD-folder.

The integration with some functions of ABCD Central, e.g. the 'ABCD MySite', is assured by the fact that EmpWeb uses the catalog(s) maintained in ABCD-Central and the specific structure of the ABCD-Loanobjects database, with all copies of a title as repeated occurrences of v959. On top of that a connection is provided by the 'bridge' folder in the Central module, where an 'end-point' web-service to a MySQL-database is available using NuSoap-PHP scripts and definitions of the queries (in config.inc.php). In this way EmpWeb can access an external MySQL users database, also storing its own transactions, rules and policies in its own SQL-tables while still accessing ISIS-catalogs. This way we think the best of both worlds (SQL and no-SQL) are combined !

### **I.3 EmpWeb vs. ABCD Central Circulation**

In the 'ABC of ABCD' manual, which serves as mainly an ABCD Central guide, an explanation is given on the differences in between EmpWeb as the 'advanced loans module' and the built-in Central submodule for Circulation. While this submodule since version 1.4 also includes reservations and MySite-functions, it is functionally equivalent to EmpWeb, using only ISIS-technology, but still with some more advanced (and often unnecessary) features of EmpWeb missing :

- the principally more efficient handling of simple transaction records in SQL
- possibility to link to SQL-stores for users on different servers and merging them with ISIS-user records
- the possibility to define easily several (even many) policies, e.g. a 'summer-' and 'exam-period'-policy, and switch in between them
- to manage the circulation system for several libraries/branches with separate or merged statistics.

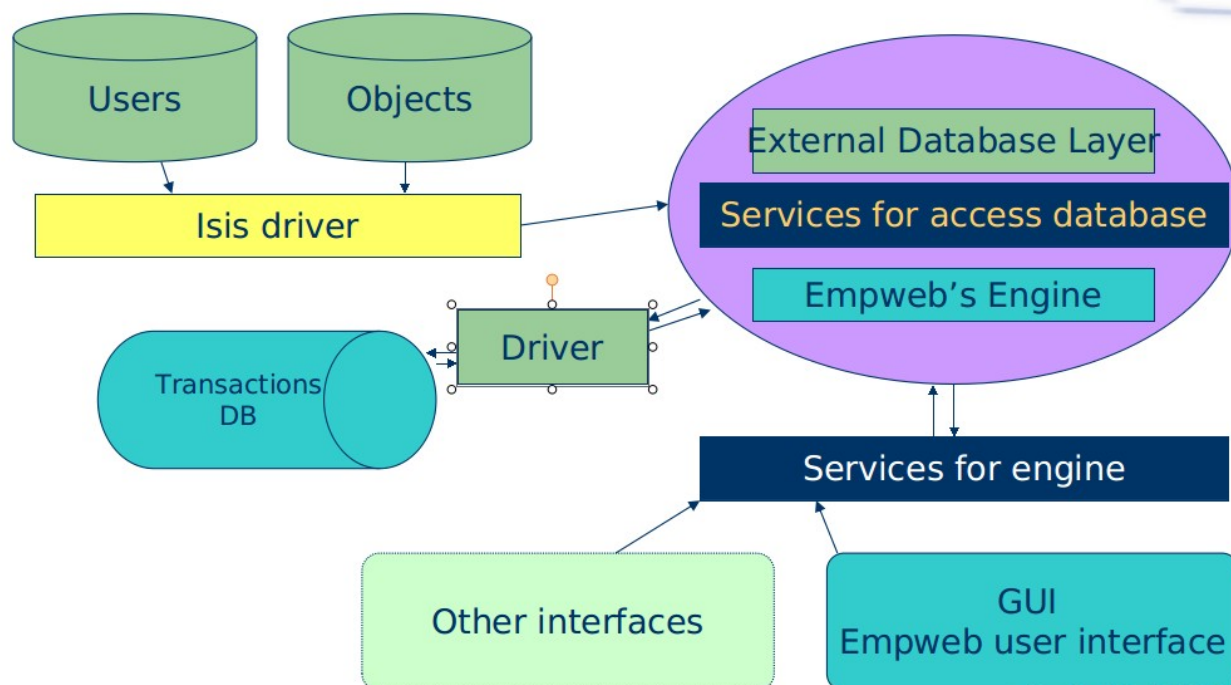
In this way EmpWeb is still the 'advanced' circulation module of ABCD and the decision to use it or not (then rather the ABCD Central circulation module) should be based on the criteria above. We can imagine that many esp. bigger organisations with more complex structure, such as university libraries, will benefit from these features while also having the necessary SQL-expertise on board. Smaller libraries however can perfectly run their circulation system based on ABCD-Central.

For further reading we refer to our paper published in Program [ref].

### **I.4 EmpWeb architecture and technical aspects**

EmpWeb is actually a rather 'generic decision-making engine' which acts as a tool to make

decisions (yes/no) based on rules implemented into the system, in our case as rules to yes or not allow and perform a transaction of a library loans-system : issuing/returning/renewing/reserving a book and dealing with fines and suspensions. The 'architecture' of EmpWeb is illustrated below :



The 'ISIS driver' is actually the CISIS-utility 'mx' which is called upon with specific commands selecting catalog-records and joining in them information from the loanobjects-database. As already mentioned, also ISIS-userdatabases can be accessed using the same method. EmpWeb's proper databases (or tables in fact) with transactions and rules/policies are accessed by one of the JDBC-drivers available, meaning most SQL-databases (MySQL, Oracle, MSSQL, PostGres...) are in the picture. The 'GUI' is styled along the ABCD-system's Cascading Style Sheets, with additional 'plugins' into the ABCD iAH (OPAC) to allow checking the availability of copies of a title and if needed reserving them through the 'MySite' function of ABCD. The use of this GUI as an ABCD-module will be discussed later in this manual but is intended to be quite easy and efficient of course. Other interfaces to the engine, as indicated into the picture above, are possible but remain outside the scope of this manual.

Technically EmpWeb is a Java servlets-based application, running on a Java-servlet server like Jetty or TomCat, and using a typical SQL-database through one of the JDB-connectors. But it can also access, through web-services, ISIS-databases. In fact EmpWeb will use the ISIS-catalogues and, optionally, the ISIS-users databases along with the SQL-tables for users (through web-services) and the transactions themselves. Also the rules and policies are all stored into SQL-tables.

Ideally therefore quite some SQL-related skills are present in the organisation when using EmpWeb. E.g. to make back-ups one has to know where the SQL-tables are stored, how to compact SQL-databases, how to export them to statistics-analysis software etc. Interfaces such as the web-based PHPMyAdmin tool can make this type of management much easier if the system-manager is not comfortable with the available command-line approach of the database.

As a conclusion we repeat that EmpWeb combines the best of both the SQL- and no-SQL worlds in one powerful flexible decision-making engine.

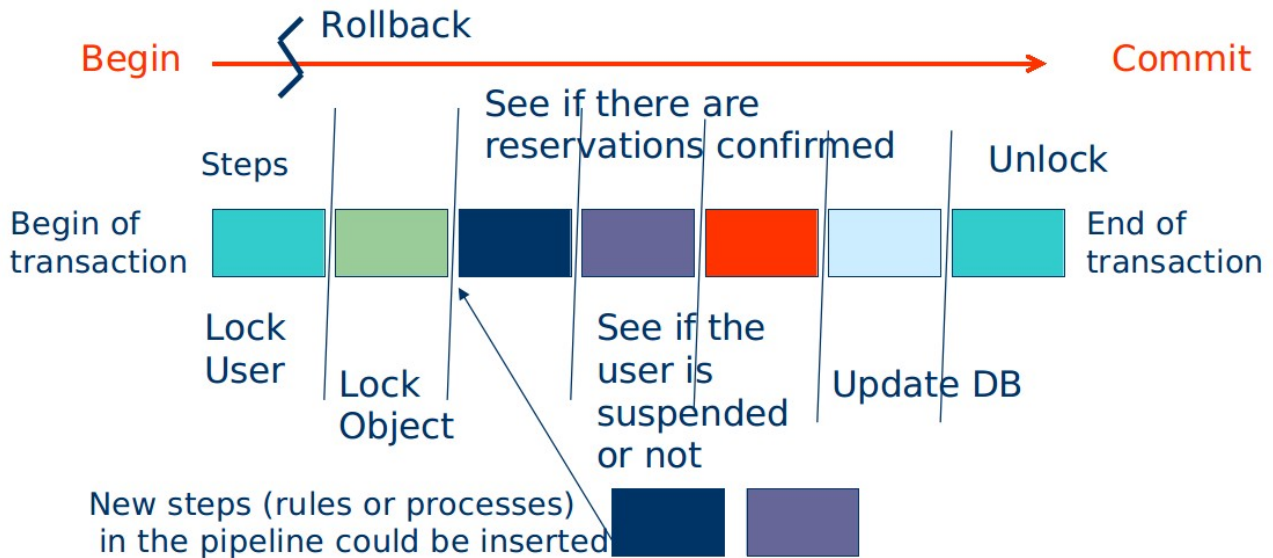
## **I.5 EmpWeb pipe-lines**

The decisions to be taken by the EmpWeb 'engine' are put in 'pipe-lines' (or sequences) of individual decisions, while at every decision the pipe-line can be exited if there is a negative decision and only at the end of the sequence or pipe-line, if all decisions were positive, the transaction will be granted and executed. E.g. when deciding on whether or not to allow a book to be issued to a user 'X', the following decisions, amongst some others, are to be considered :

- whether the user X is known to the system and if so to which user-class s/he belongs
- whether his/her status is 'active' (e.g. not suspended, not having fines to pay if it is defined as a requirement to be clear of any fines in order to loan from the library)
- whether or not the user has already reached the overall maximum allowed number of objects on loan
- whether the object (book) is known to the system and belongs to the library involved
- whether the object has copies available for loaning, and more than the defined minimum to remain in the library itself
- whether or not the available copies have not (yet) been reserved by another user
- whether or not the loans-policy (number of books, how many days etc.) have been defined for this specific combination of a user-category with an object-category
- whether or not the maximum number of loans for this specific object-category for this user has been reached
- etc...

If an EmpWeb-using library is of the opinion that it might be more efficient to first check on the status of the object and only if all is o.k to proceed with the user-'context' (as it is called in EmpWeb), then the system manager can re-arrange the sequence by shifting up the object-related decisions in the pipe-line. The EmpWeb GUI makes this a very easy exercise.

An graphical representation of such a pipe-line is given in the following illustration.



So, EmpWeb allows the system-manager to re-arrange and re-define all such sequences of decisions in the pipe-lines, thereby giving an extremely high flexibility in the system-management. System-managers who know how to use the Groovy-scripting in Java can also re-define the contents (not only the sequence) of the rules and decisions.

Here is a sample rule in Groovy code, to show it is rather 'readable'. The code actually will after getting the location (biblioteca or 'library') of the copy and of the operator, compare them and if different issue an error message to alert that the book does not belong to the actual library :

```
copyId= tc.get(TransactionContext.COPY_ID);
biblioteca= tc.getObjectValue("//hold:copy[hold:copyId='$
{copyId}']/hold:copyLocation"); transExtras=
tc.get(TransactionContext.TRANSACTION_EXTRAS);
operatorLocation= transExtras != null ?
transExtras.get("operatorLocation") : "";
if (operatorLocation != biblioteca)
{ msg.setText("This object does not belong to this library but
to :"+biblioteca);
return false;
}
else
return true;
```



However some basic pipeline 'classes' are pre-compiled into the EmpWeb system and cannot easily (without fully recompiling the whole software) be changed, e.g.

- GetUser
- ExtractUserClass
- GetObject
- ExtractObjectCategory
- GetObject Status

As an example of such non-alterable but robust fixed classes we list here the 'GetObjectStatus' code :

```
public class GetObjectStatus implements Process
{
    public ProcessResult execute(TransactionContext tc) throws
EngineException, Exception
    { String objectDb= (String)tc.get(TransactionContext.OBJECT_DB);
      String recordId= (String)tc.get(TransactionContext.RECORD_ID);
      EmpwebDB ewdb= WSBroker.getEwdb();

      ObjectStatus oStat= ewdb.getRecordStatus(recordId, objectDb);
      tc.put(TransactionContext.OBJECT_STATUS, oStat);

      // Nothing special to return, should always get OserStatus object, or
an EngineException. BBB right?
      return new ProcessResult(true);
    } // execute
} // GetObjectStatus
```

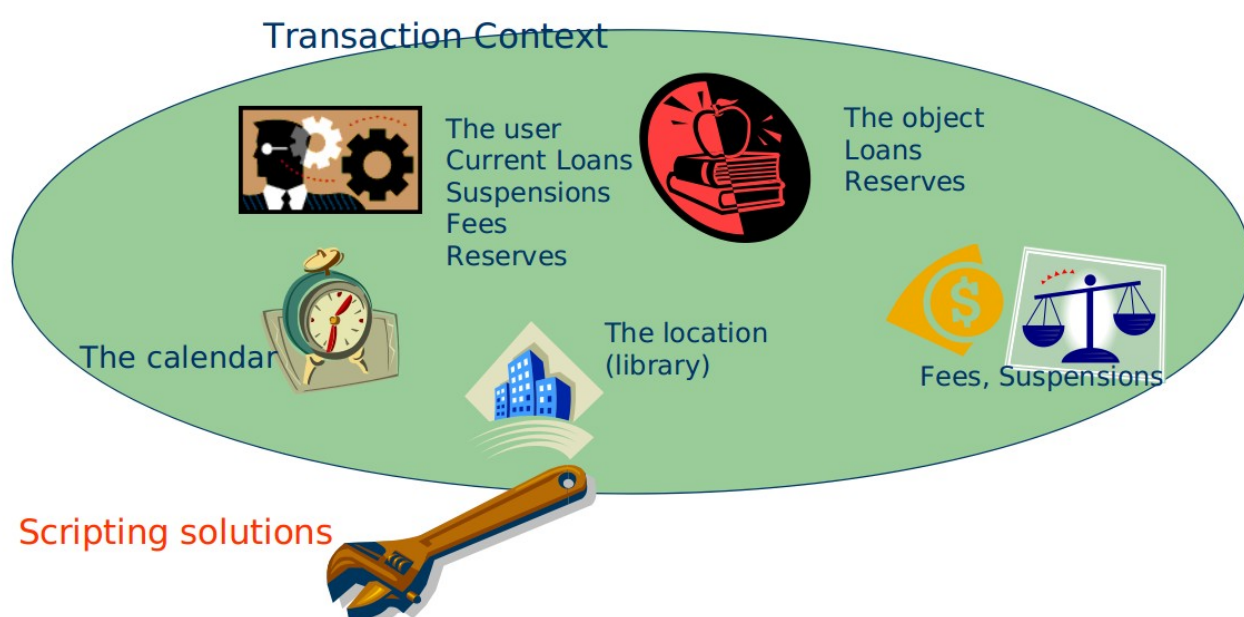
An example of a Groovy editable rule to check the validity of a user :

```
<rule class="net.kalio.empweb.engine.rules.GetUser" name="GetUser">
  <doc>Get User DOM from (userId, userDb)</doc>
  <params>
    <!-- checks for expired or disabled user -->
    <param name="checkValidity">true</param>
  </params>
</rule>
```

## I.6 EmpWeb policies

For any combination of user- with object-categories a 'profile' can be created with values for the circulation-parameters (called 'limits' in EmpWeb); a set of a number of such profiles is a 'policy' and any number of policies can be created, e.g. one for each specific period/season in a university library. By doing so, a 'third dimension' (time) is added to the two-dimensional space of user/object tuples.

A policy has to take into account several 'contexts' – technically treated as 'objects' in the scripts :



Complex policies :

The Engineering library (ING acronym) is included in the summer policy that we designed previously, but for the user class “Co-ordinators”, it will loan BKS for 20 days. Please, control the validity of the users previous to make any operation because if the expiration date is blank, Empweb is not doing actually any control!

The Agronomics library (AGR) does not admit to make on-line reservations for users which validity expire in 6 months or less. All the libraries in the system agree that if a confirmed reservation is cancelled or expired a suspension and/or a fee should be applied. If there are < 2 reservations for the same title will apply a fee of \$10, if there are >=2 reservations, will apply the fee and a suspension of 5 days.

When translated into script-rules :

## *The ABCD EMPWEB manual*

```
If object.library = 'ING' and user.type='Cordinadores' then loan.days=20
    @loan pipeline.
If user.validity =="" or user.validity is null then transaction.stop    @
loan,reservation pipelines
If month_difference_between (today,user.validity)<=6 and
transaction.comesfrom=='web' then transaction.stop.  @reservation pipeline
Create_new_fee();
If reservations.count(>)>2 then Create_suspension()    @cancelreservation
pipeline
```

Then still more complexity can be added, e.g.

All the libraries in the system agree that if a confirmed reservation is cancelled or expired, if there are < 2 reservations for the same title a fee of \$10 will apply, if there are >=2 reservations, a fee and a suspension of 5 days will apply.

At any time the libraries can agree to 'review' this rule, e.g. :

All the libraries in the system agree that if a confirmed reservation is canceled or expired, a suspension and/or a fee should be applied. If there are < 3 reservations for the same title will apply a fee of X, if there are >=3 reservations, will apply the fee and a suspension of Y days.

In a scheme then the parameters can be shown as follows :

	BKS	CDR
Coordinators	X=5 / Y=7	X=7 / Y=8
Directors	X=3 / Y=4	X=2 / Y=3

## II. Installation of EmpWeb

This manual supposes that Java JDK (not JRE) and a SQL-database (e.g. MySQL) have been installed already, using their own manuals and configuration setting tools. So we refer to the amply existing help documents on installing these and will focus only on the EmpWeb requirements proper.

### II.1 The EmpWeb launcher

In order to not only launch EmpWeb but also give it its main configuration parameters, in both Windows and Linux a script is used : empweb.sh (Linux) or empweb.bat (Windows). There is no need to understand this script, with its many typical scripting 'tricks', in detail but some useful information about this script is given here.

The script has to be invoked not only by its name but with one mandatory parameter : 'start', 'stop' being the most relevant ones. Without such parameter an error will be shown.

In fact this script will start the EmpWeb system as a Java-application, with a command to run Java followed by (quite some) parameters.

The actual 'run-command' , invoked towards the end of the script, is defined as :

```
RUN_CMD="$JAVA $JAVA_OPTIONS -jar $JETTY_HOME/"start.jar" $CONFIGS"
```

so all components of the command are constructed into variables (names starting with '\$') which are defined earlier on in the script.

The '\$JAVA' variable points to the actual java-executable in your system. The \$JAVA\_OPTIONS are defined again based on elsewhere defined variables :

```
JAVA_OPTIONS="-server -DSTART=$JETTY_START -Djetty.home=$JETTY_HOME  
-Dempweb.home=$EMPWEB_HOME -Djava.util.logging.config.file=$LOGGING_CONF  
-Daxis.xml.reuseParsers=true -Dsun.net.client.defaultConnectTimeout=10000  
-Dcisis.location=$CISIS_LOCATION -Dcisis.command=$CISIS_COMMAND  
-Dabcd.url=$ABCD_URL -Dcisis.platform=$OS -Xms128M -Xmx128M -Xincgc"
```

The relevant parts – for system management – of the script are discussed below.

#### II.1.1 Embedding into the ABCD-system

First of all the EmpWeb script puts the module into the ABCD-context with some easy-to-understand variables : where is it and how to call ABCD-Central.

```
EMPWEB_HOME="/opt/ABCD/empweb"  
ABCD_URL="http://localhost:9090/"  
OS="linux"
```

### **II.1.2 The Jetty-server configuration**

ABCD comes with EmpWeb pre-configured along with its own servlets-server 'Jetty'.

Jetty is a FOSS (free) server software very much comparable to the better known Tomcat. In fact both can do the same and are interchangeable, meaning EmpWeb can also be configured to work with Tomcat in existing installations (e.g. of dSpace or Web-JISIS applications).

The Jetty server files are just in a subfolder of EmpWeb, which itself is a subfolder in the main ABCD-folder (see *infra*).

In fact nothing has to be changed into the Jetty-server configuration as it comes with the ABCD-installation files.

The Jetty-server will be correctly linked into ABCD by the start-up script of EmpWeb (empweb.sh for Linux and empweb.bat for Windows). These are the relevant lines (for the Windows-version) :

```
REM Variables used by Jetty  
set JETTY_HOME=\ABCD\empweb\jetty  
set JETTY_START=%EMPWEB_HOME%\common\etc\start.config  
set JETTY_CONSOLE=%JETTY_HOME%\logs\jetty-console.log
```

So the only line to observe is the first one, indicating the installation directory of Jetty.

### **II.1.3 Java configuration**

EmpWeb uses the 'Java Developers Kit' or JDK, which includes the 'javac' compiler, unlike the more commonly used Java Runtime Environment (JRE). EmpWeb works with both the Oracle Java JDK as with the OpenJDK, which are currently in their main version '7' (version '8' being on its way). In fact whenever the Jetty server has been restarted for some reason, the compilation of the script will be rather noticeable as the first execution of a script will take more time (seeming 'slow') while being compiled, but subsequent runs will be very fast.

The only configuration to be done is to indicate into the EmpWeb launching scripts (empweb.sh or empweb.bat) the path to the Java-system and the Java-compiler :

e.g. in the Linux version :

```
# Java variables.  
JAVA_HOME="/usr/java/jdk"  
JAVA="$JAVA_HOME/jre/bin/java"
```

or in the Windows version :

```
REM Java variables.
```

```
set JAVA_HOME="%Program Files\Java\jdk1.7.0_45\bin"  
set JAVA=%JAVA_HOME%\javaw.exe
```

### **II.1.4 The CISIS-wrapper section**

Again in the launching scripts of EmpWeb, the link to the CISIS-executable 'mx' is defined as to allow EmpWeb to indeed access ISIS-databases (for the catalogs and loanobjects, and optionally for the users-database) :

In Windows :

```
REM CISIS Wrapper  
set CISIS_LOCATION="%ABCD\www\cgi-bin"  
set CISIS_COMMAND="%mx"  
set OS=win32
```

NOTE : other than the previously mentioned parameters in the EmpWeb launching scripts, no other lines should be checked, except maybe the value for the maximum available memory for the Java-environment in the 'Xmx'-part of the Java-settings, with higher/lower values to be set according to the available memory and the priority given to the EmpWeb application :

```
set JAVA_OPTIONS=-server -DSTART=%JETTY_START% -Djetty.home=%JETTY_HOME%  
-Dcisis.os=%OS% -Dcisis.location=%CISIS_LOCATION% -Dcisis.command=  
%CISIS_COMMAND% -Dempweb.home=%EMPWEB_HOME%  
-Djava.util.logging.config.file=%LOGGING_CONF% -Dabcd.url=%ABCD_URL%  
-Xms128M -Xmx128M -Xincgc
```

We now proceed with discussing the directories of the pre-installed EmpWeb system and specifically the parts there which are prone to configuration settings.

## **II.2 Empweb installation proper**

Again as with Jetty, EmpWeb comes pre-configured with the ABCD installation files, so in principle nothing has to be changed either for Windows or Linux in case ABCD is installed in its default directly-structure.

However some 'localisation' parameters have to be checked, esp. when installing in non-standard ways.

Configuration files for EmpWeb are mostly formatted as XML-files, so they can be edited with normal text-editors such as Notepad (Windows) or nano/gedit (Linux).

The files to be observed will be discussed under here as they come in the pre-existing subdirectories of the main EmpWeb-directory. We omit the many sub-directories with the actual coded software (.jar files as Java archives) and focus only on the files which have a meaning with respect to configuration/installation.

### **II.2.1 Common**

This is the directory where some general configuration files are found. In Windows they are not that relevant but in Linux they could be moved into their dedicated own folders, e.g. the 'etc' folder contains the launching script empweb.sh in case you want it to be in the /etc folder with all software configuration (as is meant to be in Linux). You don't need to change anything here and for someone knowing XML the configuration settings are quite easy to read, e.g. in the file

'ewdbws-jetty.xml' (for : empweb database web-services with jetty) the port is set to 8085 with the lines :

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">8085</Set>
    ...
  </arg>
```

In the sub-directory 'init.d' one can find a launch-script (as can be expected from the purpose of this folder in Linux /etc) for EmpWeb. It could be copied into the /etc/init.d folder for automatic start-up of EmpWeb at boot-time.

### **II.2.2 db**

The 'db' subdirectory of EmpWeb serves to keep backups of the transaction-database (in SQL dump-format) named by the date-stamp automatically generated whenever re-initialising EmpWeb with the dedicated function in the Administration submodule. So one has to make sure that this folder is writable to the user associated with ABCD (e.g. www-data in Linux).

### **II.2.3 dbws**

This is the folder where EmpWeb keeps its 'web-services' configuration. Web-services are used to access the external users-databases while also here the configuration for the 'ISIS-driver' to access the catalog and loanobjects (mandatorily in ISIS-format) databases is defined.

Therefore there are 2 relevant configuration files in the directory dbws/WEB-INF/conf :

- isis\_ABCD\_usersconfig.xml
- isis\_ABCDmarc\_objectsconfig.xml

Remark : if other catalogs are added to the EmpWeb circulation system, they will also have their own configuration files named in the same naming-convention here, e.g. for a database 'BIBLO' one would have the file

isis\_ABCDbiblio\_objectsconfig.xml.

## The catalog configuration file isis\_ABCDmarc\_objectsconfig.xml

In such a database-configuration file the most relevant part is the part indicating the location of the files :

```
<location>/ABCD/www/bases/marc/data/</location>
  <collection>marc</collection>

<join>/ABCD/www/bases/loanobjects/data/loanobjects,959='MARC-',v1</join>

  <altlocation>/ABCD/www/bases/loanobjects/data/</altlocation>
  <altcollection>loanobjects</altcollection>
  <altjoin>/ABCD/www/bases/marc/data/marc,1/958='CN_'v1</altjoin>
```

As can be seen, the 'collection' attribute is the reference to the ABCD catalog database (e.g. marc). The join-attribute explains how mx (the CISIS-executable delivering the reply to the webservice request from EmpWeb to ISIS) can link from this catalog-database to the loanobjects database which holds all the objects of the loans-system. Actually mx uses the 'join=' parameter with the same data as provided here in the XML-attribute :

- the name (and path) of the database
- the field tags to be considered for inclusion in the virtual catalog records at the time of creating the join (in our case : fields 1 up to 958)
- =CN\_v1 is the exact key with which mx will identify the MFN from the joined database (loanobjects in our case), which means the join will be based on the prefix 'CN\_' followed by the value of field 1 (the control number).

Further on in this configuration file the 'mapping' of the catalog to MODS (the XML-format used in the web-services process) is explained. MODS is used to reduce any catalog structure into the internal EmpWeb catalog-fields; this 'reduction' is called 'mapping'. There is no need to change anything in this mapping section, unless you need to work with non-standard ISIS-database structures, but because of this mapping in principle ANY catalog-structure can be used (as is a good ISIS-principle).

One example illustrates how this mapping puts the record-identifier as being the value of the main



part (not a subfield) of v1 :

```
<xsl:template match="record">
  <mods version="3.0">
    <recordInfo>
      <recordIdentifier><xsl:value-of
select="field[@tag='1']/occ/head" /></recordIdentifier>
    </recordInfo>
```

Another example of this 'mapping' is the section to describe the author field (v100 in MARC). In this case the mapping goes to a subfield 'a' in the 'namePart' attribute :

```
<xsl:for-each select="field[@tag='100']/occ">
  <name type="personal">
    <namePart><xsl:value-of select="subfield[@name='a']"
/></namePart>
    <role><roleTerm type="text">author</roleTerm></role>
  </name>
</xsl:for-each>
```

## Adding another catalog database to the EmpWeb system

As mentioned in the introduction, EmpWeb is pre-configured to work with the ABCD MARC21 catalog database. However it is possible, fully in the general philosophy of ABCD, to add more catalogs if so desired in EmpWeb. How this is done is explained in the next section.

For each additional catalog one has to create/edit an XML file for which the name indicates the catalog name in between the name-parts 'isis\_' and '\_objectsconfig' : we use the example of a catalog named 'LILACS', so the configuration file will be named 'isis\_LILACS\_objectsconfig.xml'.

First of all we note that, as is the case for ALL ABCD-catalogs, all physical copies of the titles are registered into the common 'loanobjects'-database, with the details in the repeatable field v959.

The mapping of the fields to MODS-XML goes as follows (we only discuss the 'monograph'-section of the actual LILACS-database) :

```
<titleInfo>
  <title><xsl:value-of select="field[@tag='18']/occ/head" /></title>
  <title><xsl:value-of select="field[@tag='19']/occ/head" /></title>
</titleInfo>

<!-- ===== Informacion de origen ===== -->
```

## *The ABCD EMPWEB manual*

```
<!-- campo 62 = EDITORIAL
      campo 63 = EDICION
      campo 64 = FECHA NORMALIZADA -->

<originInfo>
  <publisher><xsl:value-of select="field[@tag='62']/occ/head" /></publisher>
  <edition><xsl:value-of select="field[@tag='63']/occ/head" /></edition>
  <dateIssued><xsl:value-of select="field[@tag='64']/occ/head" /></dateIssued>
</originInfo>

<!-- ===== AUTOR ===== -->
<!-- campo 16 = AUTOR PERSONAL - MONOGRAF -->
<xsl:for-each select="field[@tag='16']/occ/head">
  <name type="personal">
    <namePart><xsl:value-of select="." /></namePart>
    <role><roleTerm type="text">author</roleTerm></role>
  </name>
</xsl:for-each>

<xsl:for-each select="field[@tag='17']/occ/head">
  <name type="personal">
    <namePart><xsl:value-of select="." /></namePart>
    <role><roleTerm type="text">author</roleTerm></role>
  </name>
</xsl:for-each>
```

As can be seen from the code above, the title is mapped to 2 occurrences (original and English title), while authors are mapped to personal and institutional authors.

In the following line the 'logical' internal name for the catalog is defined, in this case 'objects' :

```
<modsCollection dbname="objects" xmlns="http://www.loc.gov/mods/v3">
```

For the new LILACS catalog the following statements has to be added :

```
<modsCollection dbname="objetoslilacs" xmlns="http://www.loc.gov/mods/v3">
```

In the case of the users-database giving such a logical name was done as :

```
<userCollection dbname="isis">
```

In addition, when adding a extra catalog, the following addition has to be made to the file empweb/dbws/WEB-INF/server-config.wsdd

```
<service name="LILACSObjectsService" provider="java:RPC" style="wrapped" use="literal">
  <wsdlFile>/dbws/objects/v1/empweb-objects-service-1-0.wsdl</wsdlFile>
```

```
<responseFlow>
  <handler type="java:net.kalio.empweb.ws.ReturnAnyTypeHandler"/>
</responseFlow>
<parameter name="allowedMethods" value="*" />
<parameter name="scope" value="application" />
<parameter name="className"
value="net.kalio.empweb.dbws.mxisis.EmpwebObjectsService" />
<parameter name="configFile" value="/conf/isis_LILACS_objectsconfig.xml" />
</service>
```

This informs the webservice that this source has to be included.

Finally the following lines have to be added to the file 'engineconf.xml' :

```
<base name="objetosilacs" type="objects">
  <uri>http://127.0.0.1:8085/ewdbws/services/LILACSObjectsService</uri>
  <wsdlFile>dbws/objects/v1/empweb-objects-service-1-0.wsdl</wsdlFile>
</base>
```

This informs the EmpWeb webservice about the additional catalog to be used as a source for 'objects'-data. Accordingly the list of 'objects'-databases will then also mention the newly added catalog.

This list allows either to search a specific catalog or all of the defined ones, in which case the results will be mixed (as also done for the mixing of several users-databases).

In the illustration above one can see that the first 'hit' is taken from the MARC21 catalog, the second one however from the LILACS-catalog.

For the purpose of this 'multi-catalog' feature the name of the catalog (v10) has been added in the Control\_Number (which is the identifier of catalog-records) extraction format of the FST of the loanobjects-database, to make sure each Control\_Number indexed contains the name of the catalog :

```
1 0 "CN_"v10,"_ "V1
1 0 "CONTROL_"V1
959 0 ( |IN_|v959^i| % | / )
100 0 v10, '- ',v1
110 0 (v959^i / )
210 0 if s(mpu,v10)='MARC' then (v959^i / ) fi
310 0 if s(mpu,v10)='DBLIL' then (v959^i / ) fi
```

In the specific definition file for the new catalog (LILACS), the mapping is defined, e.g. :

```
<!-- Mapeo de campos Empweb a indices invertidos de Isis -->
```

```
<query>
  <field name="copyId" db="alternate">310</field>
  <field name="recordId" db="primary">903</field>
  <field name="author" db="primary">905</field>
  <field name="title" db="primary">945</field>
  <field name="year" db="primary">960</field>
</query>
```

The copyID field is now taken from v310 (as opposed to v210 in MARC21).

Needless to state that, as in all ABCD-catalog databases, this requires a unique identifier ('control number') to be present in the catalog-records, preferably created by a field with type 'auto-incremental' (see the ABC of ABCD manual).

### the ISIS-users database configuration file isis\_ABCD\_usersconfig.xml

As in the previously discussed catalog-database configuration file, there are 2 parts : the location section and the mapping-section.

The location of the users-database :

```
<!-- Windows version
  <location>/abcd/www/bases/users/data/</location>
  <collection>users</collection>
-->
<!-- Linux version -->
  <location>/var/opt/ABCD/bases/users/data/</location>
  <collection>users</collection>
```

Note the use of 'remark' codes <!-- and --> which de-activates all text included. In the example above the Windows-version is 'commented-out' since this configuration is for the Linux-version.

As was the case before for the 'objects' database, the XSL-coding in the second part produces the correct output-format in the expected XSL-schema. E.g. to denote the database-name to be 'isis' as will be used inside EmpWeb (e.g. when selecting users-databases available) :

```
<xsl:template match="/recordset" >
  <userCollection dbname="isis">
    <xsl:apply-templates />
  </userCollection>
</xsl:template>
```

Then the mapping is done for each field in each record. E.g. to explain that the expiration date is in the main part of v 18 :

```
<expirationDate><xsl:text><xsl:value-of
select="field[@tag='18']/occ/head" /></xsl:text></expirationDate>
```

The 'user-class' (or category) is put in v10, subfield a, as follows :

```
<userClass><xsl:value-of select="field[@tag='10']/occ/subfield[@name='a']"
/></userClass>
```

In some cases, esp. when user-database have been converted from other external formats to ABCD, the user-class might not be in subfields, so then one has to change this line to :

```
<userClass><xsl:value-of select="field[@tag='10']/occ/head]"
/></userClass>
```

In the final section of this configuration file the users-query is defined, now using the internal field-codes for the EmpWeb user-records (e.g. userClass is now in v100) :

```
<query>
  <field name="id" db="primary">350</field>
  <field name="name" db="primary">300</field>
  <field name="userClass" db="primary">100</field>
  <field name="login" db="primary">600</field>
</query>
```

Other files in this directory dbws are not relevant for configuration purposes, but contain the necessary coding, e.g. the main Java 'archive' dbws.jar in dbws/lib.

## **II.2.4 engine**

The 'engine' directory deals with the non-webservices database access, i.e. the access to the dedicated EmpWeb-SQL databases and tables (transa and university).

In the (standard) directory EmpWeb/engine/WEB-INF/conf one can find the most relevant configuration file : engineconf.xml .

This configuration file has 3 sections : the transactions names referencing their own XML-instruction files, the database names with their URL's and the most important section when it comes to installation : the actual definition of the SQL-database access.

In the transactions section nothing has to be changed. A simple example illustrates how the loan-transaction is defined referring to the loan.xml file in the trans-pipes sub-directory :

```
<transaction name="loan" path="/engine/WEB-INF/conf/trans-pipes/loan.xml" />
```

In the referenced XML-file one will find all specifications of the transaction concerned. E.g the loan-transaction itself has rules like this one :

```
if (operatorLocation != biblioteca)
```

## *The ABCD EMPWEB manual*

```
{ msg.setText(" Este objeto no pertenece a la biblioteca, sino que
pertenece a la biblioteca: "+biblioteca);
return false;
}
else
return true;
```

This means that if the operator is not working in the library of the concerned book for the given transaction (in other words : the book does not belong to her/his library) an error message is given in Spanish (this is one of the few hard-coded error messages, but it can be changed here into English !) : Este objeto no pertenece a la biblioteca, sino que pertenece a la biblioteca which in English would be : this object does not belong to the library, but instead to the library : (after which the library name will be displayed to which the book belongs).

In the databases section one should only note the name given as used within EmpWeb for each database, e.g. to denote the catalog as the 'objetos' database :

```
<base name="objetos" type="objects">

<uri>http://127.0.0.1:8085/ewdbws/services/ABCDMarcObjectsService</uri>
  <wsdlFile>/dbws/objects/v1/empweb-objects-service-1-
0.wsdl</wsdlFile>
</base>
```

A similar definition is given for the two users-database (the ISIS one and the SQL one).

Finally the most relevant section is where the SQL-connection is defined. In the default example the connection to the SQL-database for the transactions is defined as a MySQL database connection (with jdbc) at the 'localhost' URL with the default MySQL port 3306 and opening the default EmpWeb transaction database 'transa' with user-name root and password 'empweb' :

```
<base name="TRANSA" type="transa">
  <uri>jdbc:mysql://127.0.0.1:3306/transa</uri>
  <user>root</user>
  <password>empweb</password>
  <schema>ew15db-schema.sql</schema>    <!-- It's a resource inside
ew15db.jar -->
  <backupDir>/opt/ABCD/empweb/db</backupDir> <!-- Please check
access rights!-->
  <poolSettings>
```

## The ABCD EMPWEB manual

```
<driverClassName>com.mysql.jdbc.Driver</driverClassName>
<minPoolSize>3</minPoolSize>
<maxPoolSize>20</maxPoolSize>
<initialPoolSize>3</initialPoolSize>
<acquireIncrement>2</acquireIncrement>
<idleConnectionTestPeriod>30</idleConnectionTestPeriod>
<testConnectionOnCheckin>false</testConnectionOnCheckin>
<automaticTestTable>ew_test_table</automaticTestTable>
<maxIdleTime>30</maxIdleTime>
</poolSettings>
<collation></collation>
</base>
```

Needless to say that here one can change the values to connect to either another JDBC-compatible SQL-database (e.g Postgres, Oracle...) at the same or an external SQL-server with other credentials.

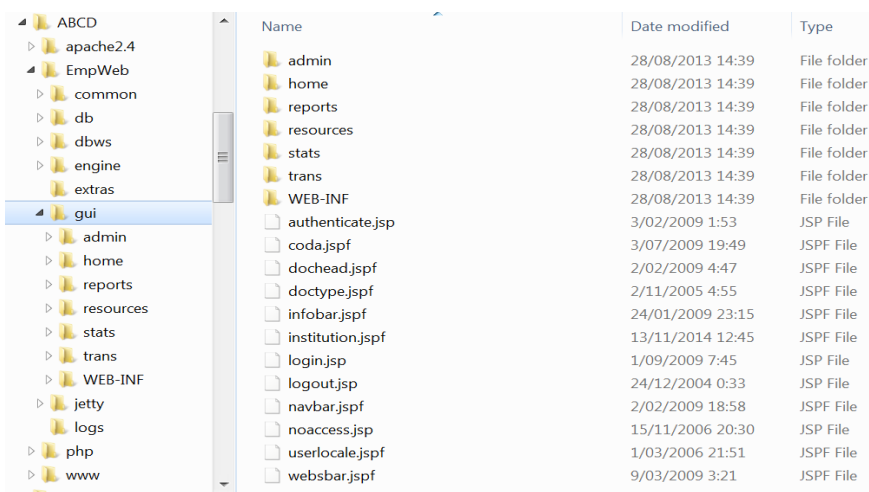
This is a very interesting feature as it shows that EmpWeb can connect to external SQL-databases where the transactions can be managed. Later on we will see also external user-databases can be accessed, e.g. tables maintained in the Registration Office of an institution.

As the login data for the SQL-database are given here, please make sure the SQL-database used will accept them.

Another important parameter here is the directory where EmpWeb will create the backups (backupDir). If this directory is not writable to the user under which ABCD is running (e.g. www-data) then simply – without unfortunately any warning ... - the 'engine' or the SQL-access won't work at all. It is a detail but an annoying one when wrong !

### II.2.5 gui

This is the directory with the actual jsp files building the Graphical Users Interface.



Name	Date modified	Type
admin	28/08/2013 14:39	File folder
home	28/08/2013 14:39	File folder
reports	28/08/2013 14:39	File folder
resources	28/08/2013 14:39	File folder
stats	28/08/2013 14:39	File folder
trans	28/08/2013 14:39	File folder
WEB-INF	28/08/2013 14:39	File folder
authenticate.jsp	3/02/2009 1:53	JSP File
coda.jspf	3/07/2009 19:49	JSPF File
dothead.jspf	2/02/2009 4:47	JSPF File
doctype.jspf	2/11/2005 4:55	JSPF File
infobar.jspf	24/01/2009 23:15	JSPF File
institution.jspf	13/11/2014 12:45	JSPF File
login.jsp	1/09/2009 7:45	JSP File
logout.jsp	24/12/2004 0:33	JSP File
navbar.jspf	2/02/2009 18:58	JSPF File
noaccess.jsp	15/11/2006 20:30	JSP File
userlocale.jspf	1/03/2006 21:51	JSPF File
websbar.jspf	9/03/2009 3:21	JSPF File

The initial JSP is login.jspf, where the EmpWeb page is built with other jsp's, e.g. relevant for localisation are :

institution.jspf where the 'header' of the page is defined with institutional data :

```
<div id="identification">
    <h1>ABCD</h1>
    <h2>Advanced Loans Module</h2>
</div>
```

This example will give 'ABCD' as the main name on top of the page, with 'Advanced Lons Module' on the second line. Changing these values will immediately have a major 'localisation' impact !

Coda.jspf defines what will happen after logging out :

```
<%
    request.setAttribute("abcd",System.getProperty("abcd.url",
"/").concat("/what.php?action=getall"));
%>
```

In this case the ABCD URL (as defined in the EmpWeb launching script) with the what.php script will be invoked. No need to change this at all.

In the WEB-INF/conf sub-directory of GUI one will find the important file 'config.properties', which exceptionally is not an XML-file but simply defines variables, e.g. :

```
gui.default_locale=en_CL_UCV
gui.languages=es_CL_UCV,en_CL_UCV,fr_CK_UCV,pt_CL_UCV
gui.hide_languages=false
```

This example is taken from the configuration for the CL\_UCV (a piloting EmpWeb user institution in Chile), putting English as the default language and defining es, en, fr and pt as the available languages. These languages are shown and selectable in the interface by putting the 'hide' parameter to false.

Many more parameters can be set here, e.g.

```
ew.hide_user_db=false
```

will NOT hide the selection list for users-databases, but obviously can also be set to 'true' to actually hide this part of the interface when doing a query or transaction.

Daring EmpWeb managers could experiment freely here without a lot of risks, as long as one remembers where one has changed what - when it is better to return to the default status.

For the actual version of EmpWeb, i.e. version 18n, the directory WEB-INF/conf/ewi18n contains the language-dependent interface elements, which therefore can be adjusted 'a volonté'.

The core-subfolder contains for the default language as well as the other defined languages the



messages on the screen, and therefore can be used to create other language versions or versions with adjusted wordings.

Each message-name is given, after the '=', the actual value to be shown, like very simply \

```
yes=Yes
```

means that in the default language (the file 'gui-properties' ) the message 'yes' will be worded as 'Yes'.

For the Spanish language this file is called 'gui\_es.properties' whereas there are also files containing messages for other modules of EmpWeb like 'engine.properties' (for the SQL-part), 'ew14dbws.properties' for the ISIS web-services part) and 'process.properties' (the transactions part).

Creating a new language version of EmpWeb would mean to duplicate all these files with the proper language indicator (e.g. 'am' could be used to create an Amharic version for Ethiopia) and translate all the values after the '=' for each message used. Then in the main 'config.properties' file the prefix for the language has to be added in order to include it into the interface for selection in the gui.languages variable where it can be added in the desired sequence of languages.

In the 'local' subdirectory of WEB-INF/conf/ewi18n one will find the same lists of language-dependent variables (or screen-messages) for the 'rules' of the loans-system (the policies), e.g.

```
loanHours=Número de horas para emprestar um item.
```

in the file 'limits\_pt.properties' will define the Portuguese message to be used to explain the 'number of hours an item can be loaned' in the rules-definition window of EmpWeb. If one understands this logic one can again translate or adjust all on-screen-visible messages for any language or indeed create a new one.

Finally there is the directory gui/WEB-INF/conf/auth where the users, the groups they belong to and the authorized actions per group are listed in XML-format.

Users.xml :

The user-data are listed for each user created in EmpWeb, including the password which is readable here if set to 'raw' (= non-encrypted).

```
<user id="egbert">
  <username>De Smet, Egbert</username>
  <password encoding="raw">egbert</password>
  <email>egbert@ac.be</email>
</user>
```

Groups.xml :

A 'group' is a set of actions with the user ID's allowed to enter :

```
<group id="home">
  <user id="admin" />
  <user id="abcd" />
  <user id="ernesto" />
  <user id="egbert" />
  <user id="albertor" />
</group>
```

defines the 'home' group (working on the main home-page) with the list of users who have access there (in the demo-installation).

Another example of a group, i.e. the query on a transaction :

```
<group id="trans-query">
  <user id="admin" />
  <user id="abcd" />
  <user id="egbert" />
  <user id="albertor" />
</group>
```

As one can see here not all defined users can do this type of action.

Properties.xml :

This file lists all users with the parameters (properties) set in their profile :

```
<user id="egbert">
  <property id="accountenabled">on</property>
  <property id="library-ARQ">on</property>
  <property id="libraryHoursUnrestricted-VET">on</property>
  <property id="libraryHoursUnrestricted-ING">on</property>
  <property id="libraryHoursUnrestricted-AGR">on</property>
  <property id="library-VET">on</property>
  <property id="connectfrom-iplist" />
  <property id="default-object-db">objetos</property>
  <property id="libraryHoursUnrestricted-ARQ">on</property>
  <property id="connectfrom-anywhere">on</property>
  <property id="library-ING">on</property>
  <property id="default-user-db">usuarios</property>
  <property id="library-AGR">on</property>
</user>
```

This file is checked whenever a user logs in and defines the further behavior of EmpWeb, e.g. if a list of IP-numbers is set (see infra, creation of operators), and the user tries to log in from a non-included IP-number, the login will be rejected (as can be seen from the resulting error URL).

## The use of 'tags' in EmpWeb GUI-folder

Most if not all EmpWeb screens use 'tags' or often-used screen elements, called from the transaction definitions and actually defined in the (sub-)folder(s of) /[empweb]/gui/tags.

E.g. the script 'record\_status\_result.jsp' calls, amongst others, the 'display' tags-folder :

```
<%@ taglib prefix="dsp" tagdir="/WEB-INF/tags/display" %>
```

in which the 'user.tag' file describes how to display the user-info, e.g. for the user-picture :

```
<table>
  <tr>
    <td rowspan="4">
      
      [etc.]
```

followed by more such html-formatted elements like user-class, contact-email, expiration-date etc.

So this is where quite some graphical elements or building blocks of the screens can be adapted if so desired.

## NOTE on the style-sheets used in EmpWeb

In the subfolder 'resources/css' of the GUI-folder the CSS-files are available. These have been adopted from the general ABCD-stylesheets (in ABCD/www/htdocs/central/css). Actually the following CSS are used (and can be changed if so desired) :

- template.css (inherited from the ABCD, importing styles y layout)
- styles.css (idem)
- layout.css (idem)
- java.css (required for the editing of Groovy scripts)
- codepress.css (required for the editing of Groovy scripts)

## Multi-linguality in EmpWeb

EmpWeb has been designed as a multi-lingual software, currently in 4 languages (English, Spanish, Portuguese and French), in which more languages can be added.

First of all the languages used are defined in the file `/[empweb]/gui/WEB-ING-conf/config.properties`, originally (first version) with Spanish, English, later on Portuguese and French were added. See the following lines of that configuration-file :

```
# web service locations
ewengine.query_service=http://localhost:8086/ewengine/services/EmpwebQuery
Service
ewengine.admin_service=http://localhost:8086/ewengine/services/EmpwebAdmin
Service
ewengine.trans_service=http://localhost:8086/ewengine/services/EmpwebTrans
actionService

# locales
gui.default_locale=en_CL_UCV
gui.languages=es_CL_UCV,en_CL_UCV,fr_CK_UCV,pt_CL_UCV
gui.hide_languages=false
```

Secondly, the actual language-sensitive messages used in the EmpWeb system are taken from the files in the subfolder `/[empweb]/gui/WEB-INF/conf/ew18n/`, where two subfolders exist : local and core.

In these subfolders EmpWeb uses text-files, named with first the module (e.g. display, engine, gui...) followed by an underscore and the language-code (except for the default 'English' where the language-code is simply omitted). All these files have the '.properties' extension.

E.g. the file 'gui.properties' in the 'core'-subfolder contains the English (since no language code) messages for the GUI-module, such as the institution-name, login/password, submit-texts and many others. The file 'gui-es.properties' contains the same messages but in Spanish.

An example from the 'local'-subfolder is the file `display.properties` where messages are translated from their internal variable-reference to – in this case – the English translation, e.g.

```
loan_info=Loan Information
operator_id=Operator
etc.
```

This means that by editing these files, i.e. changing the 'translations', one can adjust the interface with different wordings.

Adding a new language would then involve the following steps :

1. add a language-code in the file `config.properties` in the 'locales' section following the model for other languages. If one e.g. would add a 'kiSwahili' version with code SW and have it as

the default language, the lines would read as :

```
# locales
```

```
gui.default_locale=am_CL_UCV
```

```
gui.languages=am_CL_UCV,es_CL_UCV,en_CL_UCV,fr_CL_UCV,pt_CL_UCV
```

```
gui.hide_languages=false
```

2. copy all files in both the 'core' and 'local' subfolders or ONE language that will serve as the 'original' one from which to translate (e.g. English, so the files without a language-code in the naming) to the same file-names but with the language-code chosen (e.g. 'am' for Amharic).
3. Translate into the newly created files all parts after the '=' into the new language, leaving the original value before the '=' unaltered.

After this last step you have to re-load the system (restart Jetty with the 'empweb.sh' or empweb.bat script) and you will from then on have the new language available.

## III. Using EmpWeb

### III.1 System configuration

#### III.1.1 Initiating EmpWeb.

After activating ABCD\_start to actually launch Apache 'listening to' port 9090, enter the URL <http://localhost:9090/empweb/> in your browser. Remember to add the final slash to the address, or EmpWeb will not work properly.

Note that Jetty is configured to listen to port 8080 (as typical for Java Servlets), but the following lines in the Apache-configuration file for ABCD have been added to 're-route' port 8080 to 9090 :

```
ProxyPass /empweb/ http://127.0.0.1:8080/empweb/
```

```
ProxyPassReverse / http://127.0.0.1:8080/
```

After a few seconds (remember if this is the first time EmpWeb runs after the launch of the Jetty-server a compilation process will slow down the system) the main login page of EmpWeb will open.

ABCD BIREME - Centro Latino Americano e do Caribe de Informação em Ciências da Saúde  
ABCD - Empweb plug-in

Empweb Login  
| [Español](#) | [English](#) | [Français](#) | [Português](#) |

Operator Id:

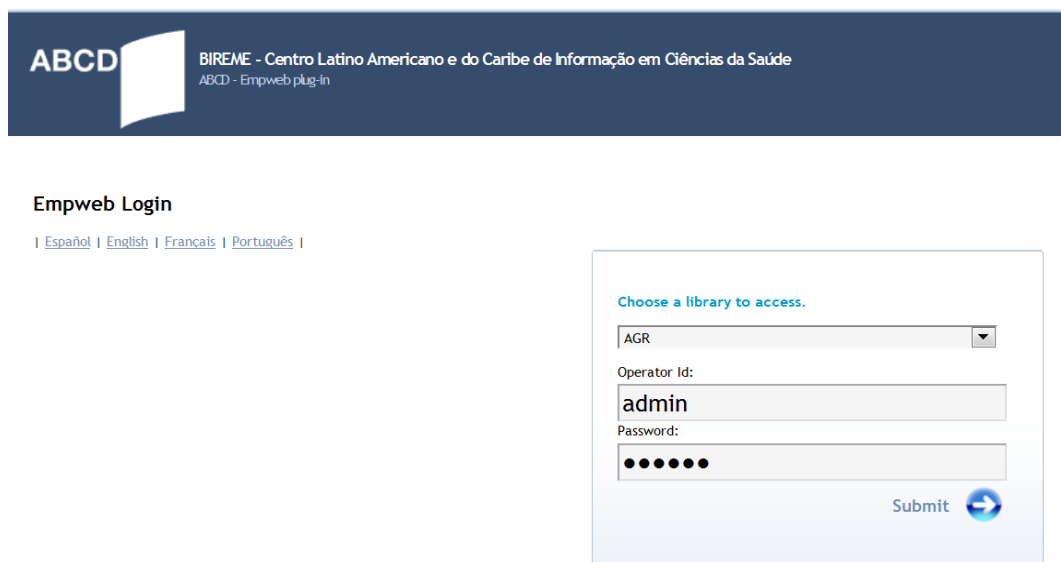
Password:

Submit

ABCD 1.4  
2014  
<http://www.abcdwiki.net>

Enter the demo administrator data (admin/empweb) and press **Submit**.

A second login page will be displayed, where you should select a library, since EmpWeb can manage a multitude of libraries or library-branches. This step is necessary only for users with access privileges to more than one library.



ABCD BIREME - Centro Latino Americano e do Caribe de Informação em Ciências da Saúde  
ABCD - Empweb plug-in

Empweb Login

[Español](#) | [English](#) | [Français](#) | [Português](#) |

Choose a library to access.

AGR

Operator Id:

admin

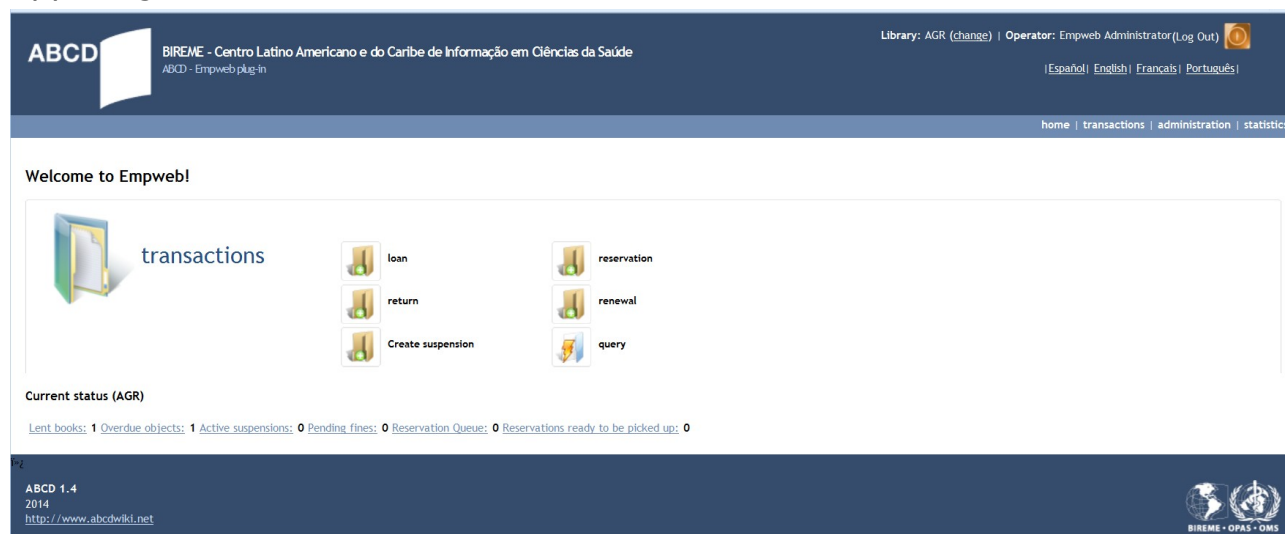
Password:

Submit

Select the library you will work in and press Submit. The EmpWeb application will now open.

This will put you into the system, with the main menu of the transactions

readily available but additional system-management functions present in the upper-right corner as links.



Before discussing the intentionally 'simple' use of the transactions themselves, we first have to discuss the many system-management issues at stake, which should typically be addressed before the system can be used indeed.

### **III.1.2 Main initialisation of the EmpWeb SQL-tables**

From here on we will assume the SQL-database used is MySQL, but generally speaking other SQL-databases won't be much different in their usage.

EmpWeb uses 2 databases in the selected SQL-environment :

- university : this database contains two tables (users and user-types) and is used to store circulation-system users with their main characteristics



(name, ID, category...) and the user-types or categories

- transa : the main database where all transactions will be stored into their proper tables.

### The university database

This database has only 2 tables, as can be shown by MySQL through the commands 'use university' (to select that database) and 'show tables' (after having logged in into MySQL-command-line modus) :

```
mysql> use university;
Reading table information for
You can turn off this feature

Database changed
mysql> show tables;
+-----+
| Tables_in_university |
+-----+
| user_type             |
| users                 |
+-----+
2 rows in set (0.00 sec)
```

To initialise the 'university' database, it is a good idea to load the structure with some sample data (for users) from the file 'university.sql', which can be found in the extras-directory of the EmpWeb installation folder. This can be done either by e.g. PHPMysqlAdmin or with a simple command :

```
MySQL> source /opt/ABCD/empweb/extras/university.sql;
```

which will show the following output :

```
mysql> source /opt/ABCD/empweb/extras/university.sql;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Now 4 demo-users are available, listed by the command

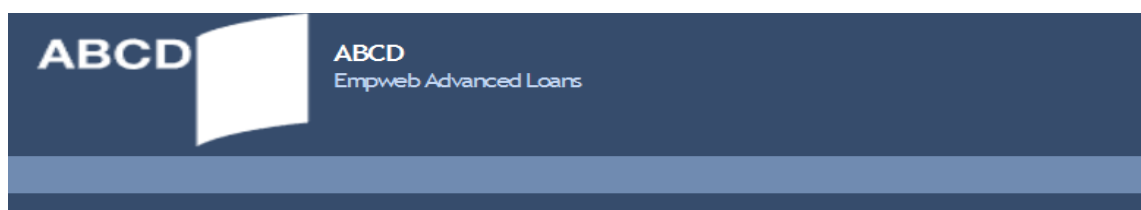
```
mysql>select * from users;
mysql> use university;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from users;
+----+-----+-----+-----+-----+-----+-----+
| id | identification | last_name | first_name | user_type_id | valid_until | login |
+----+-----+-----+-----+-----+-----+-----+
| 1 | ABX-6272 | Johansen | Arthur | 1 | 2022-12-24 | arthur |
| 2 | ABX-6362 | Summer | Albert John | 1 | 2022-10-22 | summer |
| 3 | CBX-1627 | Winter | Gladys | 2 | 2022-03-24 | gladys |
| 4 | CBR-22782 | Spring | Ernest | 4 | 2022-07-31 | spring |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Later on we will see that these 4 demo-users can be queried from within EmpWeb as one of the two available users-databases.

After accessing EmpWeb with the system-administration login, the main Transactions page will open but also giving access to the 'Administration' options, and you must perform a very important step before being able to use EmpWeb for the first time, i.e. initialising the transactions database.

In order to initialise the transactions database, go to the Administration menu, select the Databases (Bases) sub-menu and a initialisation Confirmation/Warning window will appear. Confirm the operation by clicking **YES**.



## **Confirmation**

### **Initialize Empweb databases**

This function creates new internal bases for Empweb, losing previous configurations and transaction data.

Are you sure you want to initialize all databases?

If the process is successful, you will receive a confirmation message.

## **Confirmation**

### **Initialize Empweb databases**

#### **Successful request**

Success initializing databases

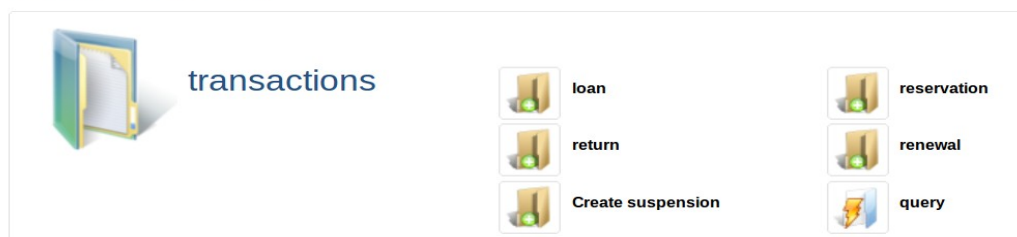
This means that the 'engine' has successfully accessed the SQL-database for storage of the transactions, requiring your SQL-database server to run and the parameters (in engine.conf.xml) to be correct. The transa-database is now available but without any data.

Initialising this database will cause EmpWeb to automatically create a 'dump' of the database in the directory defined as the 'backup-'directory (see supra) in a file named after the date/time-stamp with extension .sql. This is a backup indeed as it can also be restored into MySQL with a 'source'-command.

Back at the main Transactions page, you will now see that a 0 (null) figure has

appeared at the right of each Status item, confirming that the 'transa' database has been activated.

**Welcome to Empweb!**



**Current status (AGR)**

Lent books: 0 Overdue objects: 0 Active suspensions: 0 Pending fines: 0 Reservation Queue: 0 Reservations ready to be picked up: 0

**NOTE : Database initialisation problems.**

If you had a previous MySQL installation on your computer, in Windows the old root/password and other configuration data remains in the system (somewhere in your 'users'-folders) even if you delete the Program Files\MySQL folder completely, and may affect the functioning of EmpWeb, including the initialisation of the 'transa' database.

The solution to this problem is to open a console, invoke mysql and manually drop the database:

- 1- Open a cmd window.
- 2- Execute the command **mysql -uroot -p** (press Enter)
- 3- \*\*\* insert the password \*\*\* (type the password 'empweb') (press Enter)
- 4- mysql> DROP DATABASE transa; (press Enter)
- 5- mysql> quit; (press Enter)

After doing this, you will be able to initialise the mysql database from EmpWeb.

**NOTE : Manual configuration of EmpWeb under MySQL.**

In order to configure the root user account manually, you have to edit the file /ABCD/empweb/engine/WEB-INF/conf/engineconf.xml. Use a text editor like Notepad for this. Locate line 150 and from there on you will find the following content:

```
<base name="TRANSA" type="transa">
<uri>jdbc:mysql://localhost/transa</uri>
<user>root</user>
```

```
<password>empweb</password>
<schema>ew15db-schema.sql</schema>    <!-- It's a resource inside
ew15db.jar -->
<backupDir>C:/ABCD/empweb/db</backupDir>
<poolSettings>
  <driverClassName>com.mysql.jdbc.Driver</driverClassName>
  <minPoolSize>3</minPoolSize>
  <maxPoolSize>20</maxPoolSize>
  <initialPoolSize>3</initialPoolSize>
  <acquireIncrement>2</acquireIncrement>
  <idleConnectionTestPeriod>30</idleConnectionTestPeriod>
  <testConnectionOnCheckin>false</testConnectionOnCheckin>
  <automaticTestTable>ew_test_table</automaticTestTable>
  <maxIdleTime>30</maxIdleTime>
</poolSettings>
<collation></collation>
</base>
```

Change <user> and <password> and save the file. For the changes to have effect, you must shut down ABCD with ABCD\_exit.bat, and then restart using ABCD\_start.bat

At this time it might also be a good idea to check the value for the backupDir-variable : if that directory (or 'folder' in Windows) is not writeable for the EmpWeb-user, the initialisation will – without any useful error-message – not work !

### III.1.3 Configuration of libraries

The steps to configure libraries in EmpWeb are the following:

1. Change the library definition in the **conf-getLibraries pipeline**.
2. Change global parameters of the library in the **globalenvironment pipeline**.
3. Assign user rights to access these libraries (or branches).

#### ***Change the library definitions in 'globalenvironment' pipeline***

After starting the EmpWeb application, select Administration -> Pipelines and a

screen appears with the title Transaction Pipelines Administration (similar to the figure below) where you will see the first two options.

In the first option the system-administrator has to define the libraries or branches which will be administered by EmpWeb. The interface actually provides a text-editor on the file '[empweb-home]/engine/WEB\_INF/conf/conf-pipes/globalenvironment.xml', which – as is the case with other such configuration text-files - is mostly consisting of comment-lines (starting with '//') to explain the use but towards the end contains the most relevant section where the libraries and for each of these the latest time at the day when short-loans can be initiated (maxHourForLoanByHour) are defined :

### Edit Transaction Pipeline

#### Pipeline

Name                    globalenvironment  
Type                    configuration  
Evaluation Method   shortcut  
Classpath               /engine/WEB-INF/trans\_rules/classes/

#### Process and Rules

Enabled	Type	Name	Description	Actions
<input checked="" type="checkbox"/>	process	DumpEnvironment	Return this pipeline's environment section as XML	Up   Down   <a href="#">Edit</a>   <a href="#">Delete</a> <a href="#">Create new process</a>

#### Environment parameters

-->

```
<param name="pucvua.ING">ING</param>
<param name="pucvua.ARQ">ARQ</param>
<param name="pucvua.AGR">AGR</param>
<param name="pucvua.VET">VET</param>

<param name="maxHourForLoanByHour_ING">1900</param>
<param name="maxHourForLoanByHour_ARQ">1700</param>
<param name="maxHourForLoanByHour_AGR">2359</param>
```

XML

In this example we can see that the libraries ING, ARQ, AGR y VET are configured. The part of the library names 'pucvua.', preceding the actual library name abbreviation, is inherited from the piloting institution, but only the actual names will be used for selecting a library later on.

Some other 'hours' can also be defined, optionally, for each library, e.g. the starting hour for reservations or the latest hours for returning books, for each library individually.

By default in EmpWeb, 15pm is configured to be the limit for returns at the date due; if preferred this time-table can be adjusted from the same menu Administration, Pipelines and this configuration-file. Returns processed from 15p.m. will be considered as late and subject to sanctions (fines or suspension) according to the user profile.

## Change the library definitions in conf-getLibraries pipeline

In the second pipe-line option 'Conf\_getLibraries' two additional characteristics are defined for each library in the network : the IP-numbers (or ranges, like e.g. '\*' - for any IP-number – in the example below) and the opening-hours.

In the illustration below this is again done for 4 library-branches of a university-library : ING, ARQ, AGR and VET.

### Edit Transaction Pipeline

#### Pipeline

Name	conf-getLibraries
Type	configuration
Evaluation Method	shortcut
Classpath	/engine/WEB-INF/trans_rules/classes/

#### Process and Rules

Enabled	Type	Name
<input checked="" type="checkbox"/>	rule(Script)	GetLibrariesFromEnvironment UCV - Obtien

Submit

#### Environment parameters

```
<environment>
  <param name="libraryIp_ING">*</param>
  <param name="libraryIp_ARQ">*</param>
  <param name="libraryIp_AGR">*</param>
  <param name="libraryIp_VET" />

  <param name="libraryHours_ING">08:00-18:00</param>
  <param name="libraryHours_ARQ">09:00-17:00</param>
</environment>
```

XML

Submit



## **Assigning user rights to the system.**

After defining the libraries which populate the system in EmpWeb, you must establish the necessary permissions for the users to access EmpWeb. The permissions will only have effect as of the next session.

Go to the EmpWeb Administration menu - > Operators. Select the “super user” with access to all applications - in this case the user **abcd** who has full administrator privileges.

From this access you can create new users and assign the corresponding rights. It is also possible to edit, modify or delete existing users, as shown in the next image.

By editing the system-administrator **admin** (case-sensitive !), you gain access to the library definitions and can assign the new access permits to each.

**Warning:** Remember to save the changes clicking on the Submit button for each group, **or the new settings will not have effect.**

Once you have assigned the new access permissions to the libraries, you must logout from the system (without closing EmpWeb).

Login again, and now you will see the list of available libraries assigned to the current user/password.

### **III.1.4 Administration of calendars in EmpWeb.**

The administration of calendars for the loan transactions and returns in EmpWeb is global for the whole system and all libraries, so we can define all the calendars for each year. For this we need to enter into the option Administration / Calendars.

When editing a calendar, a frame is presented in which the rows correspond to months and the columns to the days.

Ticking the boxes for each day of the month we can define the non-working days to be taken into account by the system. Each holiday or non-working day will be skipped when the system calculates the latest return-day for a loan or pick-up date for a reservation.

Without having defined the calendars, no transaction will be granted since then the system has no information on these days-to-skip !



## Calendar Administration

### Edit Calendar

Calendario de 2009 | Days not counting

	Día del mes																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Enero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Febrero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Marzo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Abril	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Mayo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Junio	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Julio	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Agosto	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Septiembre	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Octubre	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Noviembre	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Diciembre	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

In the case that a non-working day had to be inserted in the calendar, all late returns on that date will be automatically transferred to the next working day.

### III.1.5 Creation of new operators.

It is crucial to analyze correctly the operations assigned to each EmpWeb operator in order to assign the necessary permissions which give access to the different databases and operations.

After entering with the system-administrator login data a second window is presented for selection of the library within which will be worked.

**Warning:** to create or modify operators one has to take into account the necessary permissions for such operations; in this case the user admin/empweb has the adequate permissions.

From the function administration, we can access the menu 'operators' which lists the operators already defined and offers the functions for editing/copying/deleting existing users on top of the creation of new operators. Such a list is shown in the picture below, along with all operations (presented as links) which can be performed on each operator (or staff-member) : edit, copy or delete.

## Operators Administration

### Operators List

<u>Operator Id</u>	<u>Operator Name</u>	<u>E-mail</u>	<u>status</u>	<u>Actions</u>
admin	Empweb Administrator	root@localhost		<a href="#">Edit</a>   <a href="#">Copy</a>
ВПАДИМИР	Vladimir the Russian		Disabled	<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>
شريف	Omar Sharif		Disabled	<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>
abcd	Administrador ABCD	abcd@abcd.org		<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>
ernesto	Spinak, Ernesto	spinaker@adinet.com.uy		<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>
egbert	De Smet, Egbert	egbert@ac.be		<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>
albertor	Rosemberg, Alberto Alejandro	arosemberg@gmail.com		<a href="#">Edit</a>   <a href="#">Copy</a>   <a href="#">Delete</a>

[Create New Operator](#)

ABCD 1.4  
2014  
<http://www.abcdwiki.net>

### Creating a new operator

When creating a new operator, the first thing requested is the operator's ID to be used for login into EmpWeb; subsequently a window will be presented with several sections.

These sections are :

- 1- information on the operator;
- 2- IP from where access will be granted;

- 3- libraries which can be accessed and their operating hours,
- 4- links to operations and permissions;
- 5- and finally the properties or permissions to search in the user and loanobjects databases.

Each section has a button SUBMIT to be used each time permissions in the section have been defined.

After the creation of an operator has been confirmed, we still need to enter the details of the operator account, the IP's authorized to enter from and the assigned permissions.

Defining the IP's from which the operator can access, adds additional security to the one based on the login data (login/password) since the user/operator will only be allowed to enter from this specific library or IP. Also it is possible to enter an IP from which the operator can access in the List of IP Addresses or to enter more addresses separated by a semi-colon (;).

The access 'Anywhere' means allow access from any computer on the internet. In the following section of the operator's profile we need to define on which library an operator has rights to operate and whether operations are confined by the operating hours table or not.

#### **Membership of the operator groups and permissions assignment**

In the next section we have to indicate which are the actions for which the operator has permission. The action 'enter Home' always needs to be permitted as without it an operator cannot do anything.

In the last section of the permissions window (see next picture) the database(s) of loanobjects and users which can be administered are to be indicated. At the time of realizing a transaction access to one or all databases will be needed.

<input type="checkbox"/>	admin-status	/ administration / status
<input type="checkbox"/>	admin-policies	/ administration / policies
<input type="checkbox"/>	admin-exceptions	/ administration / exceptions
<input type="checkbox"/>	admin-calendar	/ administration / calendars
<input type="checkbox"/>	admin-pipelines	/ administration / pipelines
<input type="checkbox"/>	admin-pipelines-engines	/ administration / pipelines / ?
<input type="checkbox"/>	admin-pipelines-languages	/ administration / pipelines / ?
<input type="checkbox"/>	admin-operators	/ administration / operators
<input type="checkbox"/>	admin-bases	/ administration / bases
<input type="checkbox"/>	reports-status	/ reports / status
<input type="checkbox"/>	reports-hist	/ reports / historic
<input type="checkbox"/>	reports-letters	/ reports / cartas
<input type="checkbox"/>	webproxy	/ webproxy
<input type="checkbox"/>	webproxy-ciro	/ webproxy / ???ciro???
<input type="checkbox"/>	webproxy-users_photo	/ webproxy / ???users_photo

**Operator properties**

Default Object Database

Default User Database

As a result of such operator profile definition, when an operator tries to enter a transaction for which the necessary authorisation has not been given, EmpWeb will show an error message informing that the operator is not properly authorized.

### III.1.6 Linking the ABCD-users and Loanobjects databases.

The user-databases, be it the externally managed 'SQL' user-tables or the internally managed ISIS users-database, are shared by all libraries in EmpWeb; when having entered as a system administrator it will be possible to consult the existing users in both databases either as a specific 'query' in the dedicated menu or when clicking on 'search' at a transaction screen. The actual query screen of EmpWeb allows querying users, objects and transactions; in the illustration here already a search for 'all users' is filled in by using the ISIS-truncation mark '\$' without anything else (which is a bad idea when dealing with large user-databases !), in this case to illustrate the list of sample-users from both the SQL and ISIS-databases :

#### User Query

User ID:	<input type="text"/>
Name:	<input type="text" value="\$"/>
Database:	<input type="button" value="Search All"/> <input type="button" value="Search User"/> <input type="button" value="Reset Form"/>

#### Object Query

Record Id:	<input type="text"/>
Copy Id:	<input type="text"/>
Title:	<input type="text"/>
Author:	<input type="text"/>
Database:	<input type="button" value="Search All"/> <input type="button" value="Search Object"/> <input type="button" value="Reset Form"/>

#### Transaction Query

Transaction Id:	<input type="text"/>
<input type="button" value="Search Transaction"/>	

When clicking on 'Search User' (in 'Search all' databases mode), the following sample users will be shown :

<u>Name</u>	<u>User Class</u>	<u>User ID</u>	<u>User expiration</u>
Johansen, Arthur	Students	<a href="#">ABX-6272</a>	12/24/22 12:00:00 AM
Summer, Albert John	Students	<a href="#">ABX-6362</a>	10/22/22 12:00:00 AM
Winter, Gladys	Professors	<a href="#">CBX-1627</a>	3/24/22 12:00:00 AM
Spring, Ernest	Post-graduate Students	<a href="#">CBR-22782</a>	7/31/22 12:00:00 AM
Guilárte, Angela Margarita	ad	<a href="#">01</a>	12/12/10 12:00:00 AM(user expired)
Tanio Reyes, Josefa	di	<a href="#">02</a>	12/12/11 12:00:00 AM(user expired)
Azuaje, Rafael	ad	<a href="#">03</a>	12/12/11 12:00:00 AM(user expired)
<b>Number of results: 7</b>			

Looking at this list more carefully, one can note that :

1. 4 users from the SQL-users table are listed
2. 3 users from the ISIS-demo users-database are listed, but with expiry dates already passed.

In this case no ISIS-users can be served, as they will only be taken into account when they are still

'valid' according to their expiry-date. So the ABCD Central users-database has to be edited in order to update the actual expiry dates

The system manages transactions based on the user ID, so that if the user ID is known (e.g. by scanning the barcode on the ID) we enter it into the dedicated field and there is no need to perform a search. Such could be done by typing or reading by barcode reader – or any other similar device.

When the user ID is not known, a search is to be done using the ISIS-search technology; for this we enter the first characters followed by the ISIS-truncation (\$)m, e.g. 'guilart\$', and pressing the button 'search' will locate the user(s) who fit the search expression.

Any record fitting the search expression will be displayed; the correct entry will be selected by clicking on the user ID field and this then will be passed to the loans form for further processing.

### **Create a new user.**

For creation of a new user from ABCD we need to enter the Database Administration module (ABCD Central) and select the users database (a demo database is included with the EmpWeb distribution).

The option Data Entry allows to enter a new user in the database with the following fields :

- User type (based on the list of user types defined in Usuarios.TAB in the folder DEF of the user-database)
- sex male/female,
- expiry date (refers to the date until which the user will be considered active);
- expiry date in ISO date format (automatically created by the system)
- user code;
- Names;
- ID-card;
- Affiliation
- campus/branch
- Adm. Level 1
- Adm. Level 2
- Physical Address
- Mail Address
- City

- Country
- Telephone
- Fax
- E-mail
- Login to access Mysite
- Password to access Mysite
- Picture 4x4

In this distribution of EmpWeb the user\_ID is extracted from field 10 from the user records or a SQL table called user\_type which is linked e.g. from MySQL. It is possible to configure the link to the users database from any field or combination of fields where the data can be extracted.

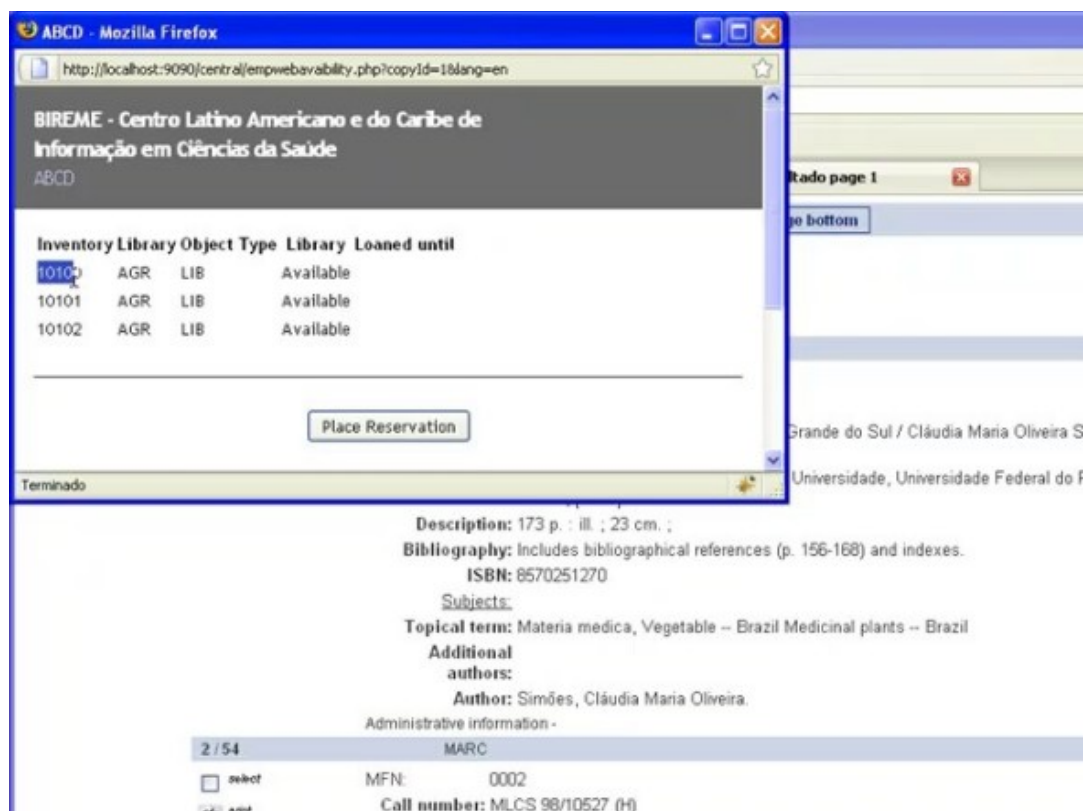
Editing the user record in ABCD-Central also allows uploading a picture to be included in the user-record. When we proceed with saving the record, by clicking on the 'diskette' icon at the bottom of the form, the user will immediately be activated and can loan materials from the library to which he is linked.

When we access user-data from EmpWeb, clicking on the ID field it can be seen that the complete profile includes the picture plus the active loans and other information.

### ***Loanobjects and the loanobjects database***

The loanobjects in this EmpWeb distribution are linked to the MARC bibliographic database and when we view the records we can know whether there are copies available by clicking the button which is shown at the left of each record.

This action of consulting item availability, will allow making reservations; at this stage the user will have to enter his/her login data for the reservation registration.



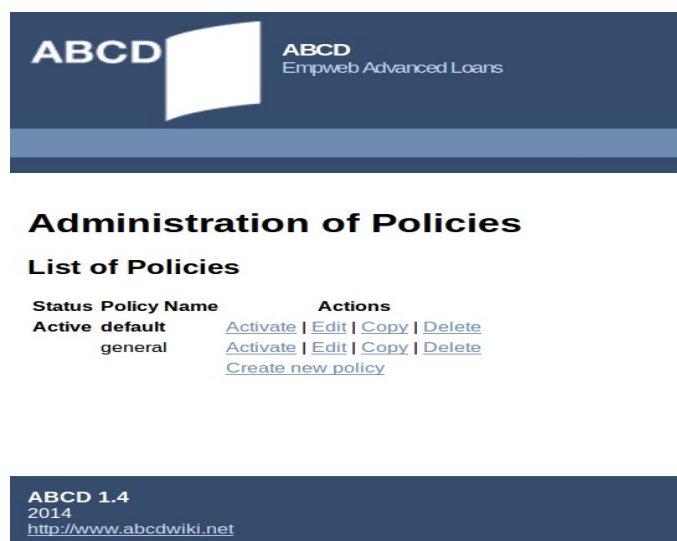
This database LOANOBJECTS has a field 'Control Number' which links the record with the bibliographic database (MARC); also there is a field to identify the name of the bibliographic database (as there can be many in one Loans system) and finally there is a repeatable field in which the data of each copy are entered into the respective subfields for ID, library, shelf, part and volume no. and object type (subfields i, l, b, v and t).

To make a return transaction is simple : we enter the Transactions module and put the ID of the copy to return; when clicking the button 'process return' some information about the success (or not) of the return transaction will be given and this will immediately be reflected in the item status of the library and the information of the MySite fuction.



### III.1.7 Definition of policies

Policies as basically sets of profiles, of which only one can be 'active'. A good practice is by starting with a 'general' or default policy with which the library circulation system can start operating, then as needs or ideas to make the system more sophisticated, to add new policies and activate then whenever needed.



### III.1.8 Definition of profiles by user type and object type.

To define the treatment which EmpWeb has to apply to each user type and loanobjects type, profiles are used.

A policy involves a set of profiles and each profile is a combination of user type with object type.

When editing the current policy, we will see that in reality in the installation package contains only one profile only which is applied to any object type and any user type, as indicated by the asteric (\*), for the time being.



## Policy Information

### Edit Policy

#### Policy details

Policy Name default  
Policy Id 20090127\_0A6C20

#### Profiles Matrix ([show as list](#))

	User Class
Object Category *	<a href="#">Edit</a>

[Create new profile](#)

When we edit a profile, by clicking on the link 'Edit objects category' (as seen in the previous illustration), we can see all the parameters applied in each transaction.

There are quite many such parameters indeed, again showing the 'advanced' idea of EmpWeb as a circulation manager. Not all of them have to be used and some can therefore be left empty.

### ***Components of a profile.***

A profile is based on four columns : the name of the parameter, the value of the parameter, the description and the transactions affected by this parameter.

In the profile currently being edited, applied to all user types and object types, we can observe the following based on the four columns :

- no fine created when a return is late (False)
- fine created when a user cancels a confirmed reservation (True)
- no fine created when a user has a reservation which expires (False)
- suspension created when an item is returned late (True)
- no suspension is created when a user has a reservation which expires (False)
- the maximum number of days to wait for a user to collect a reserved item is put at 3

- fine amount for a confirmed reservation not taken (5)
- (multiplier of) the fine when a reservation expires and the item has no other reservations(1)
- fine amount when a reservation expires and other users have reserved the same item (2)

Summarizing this section defines all parameters and limits applied on each of the transactions.

For example, the number of days for a loan for any object and user type is put at 5 in the last parameter shown in the illustration below :

## Profile Information

### Edit Profile

#### Profile Details

User Class \*

Object Category \*

Date 3/16/09 4:29 PM

Policy Name [default \(20090127\\_OA6C20\)](#)

#### Limits

Name	Value	Description	Used in transaction
createFineIfLate	<input type="text" value="false"/>	(Boolean) Enables creation of fine when an object is returned late.	return
createFineIfReservationConfirmedIsCancelled	<input type="text" value="true"/>	(Boolean) Create a fine if a confirmed reserve is cancelled.	
createFineIfReservationExpired	<input type="text" value="false"/>	(Boolean) Create a fine if the user has an expired reservation	cancelreservation
createSuspensionIfLate	<input type="text" value="true"/>	(Boolean) Enables creation of a suspension when an object is returned late.	return
createSuspensionIfReservationExpired	<input type="text" value="false"/>	(Boolean) Create a suspension if the user has an expired reservation	cancelreservation   cancelwait
expirationDays	<input type="text" value="3"/>	Number of days to wait for a user to pick up a reservation (before expiring)	wait   return   reservation   cancelwait
fineAmountIfConfirmedReservelsCancelled	<input type="text" value="5"/>	Fine amount for a cancelled confirmed reservation.	
fineAmountIfExpiredNormal	<input type="text" value="1"/>	Fine amount (multiplier) for an expired reservation when the object has no reservations	cancelreservation
fineAmountIfExpiredReserved	<input type="text" value="2"/>	Fine amount (multiplier) for an expired reservation when other users have reservations for this object	cancelreservation
fineAmountNormal	<input type="text" value="20"/>	Fine amount (multiplier) when an object is returned late but has no reservations or waits.	return
fineAmountReserved	<input type="text" value="30"/>	Fine amount (multiplier) when an object is returned late and it has reservations or waits by other users.	return
loanDays	<input type="text" value="5"/>	Number of days to lend an object. (May be library days or absolute days, as defined in the transaction)	wait   loan   reservation   renewal

## Creation of a new profile

When we need to define a new profile of a policy, we need to enter Administration – Policies, Edit policy (in this case Active Default, Edit, Create New Profile) and in there a new profile can be created.

E.g. when we want to create a new profile for the user type 'coordinators', we must identify the user category as 'coordinators' and the object type (e.g. CDROM).

In the window we must define each of the parameters and limits until all values are defined with TRUE or FALSE or a number.

E.g. to illustrate again the use of circulation rules parameters we list all of them for a specific profile :

### *The ABCD EMPWEB manual*

- no fine created when an item is returned late or no fines used at all (False)
- suspension created when an object is returned late (True)
- no suspension is created when a user has a reservation expired (False)
- number of days during which the user can collect the reserved item is put at 3.
- Fine amount (multiplier) when an object is returned late without other reservations : 0.
- Fine amount (multiplier) when an object is returned late and other reservations are noted : 0
- The number of days the loan lasts for this object (which can be absolute days or library days, as defined in the transaction). In this case put at 15.
- Total maximum fine a user can accumulate without losing the possibility to loan. Here it is 0 since no fine system is used.
- Maximum number of fines a user can have without losing the right to loan. Left at 0 here.
- Maximum number of fines a user can have without losing the right to make reservations : 0
- Maximum number of loans allowed when user has fine : 0
- If the user has late returns how many new loans will be allowed ? Leave at 0
- Max. Number of loans allowed when user has suspension. Left at 0 implies that when suspended a user cannot have loans anymore.
- Maximum number of loans of the same object category, put at 2, meaning only two CDROMS can be taken at the same time
- Maximum no. of loans for the same volume/part item, put here at 1
- Maximum no. of loans this user can have in total, here put at 7
- Maximum no. of renewals for this user, put at 3, meaning that 3 renewals are allowed, but from MySite only 2 will be allowed.
- Maximum number of reservations if the user has fines, so in this case 0 since no fine system is used here.
- Maximum number of reservations if the user has late returns : no reservations are allowed with delayed returns
- Maximum no. of reservations if the user has suspension running, 0 means that the a suspended user cannot take any reservation
- Maximum no. of reservations for objects of this category. Same as the value for loans and same for reservations of the same type of objects
- Maximum no. of reservations for the same record, same case of 1 for loans would be the same case applied for volume/part. [???
- Maximum no. of reservations allowed in total, in this case put at 5
- Suspension days when a reservation expires, in this case we do not apply suspension days, so 0 whether there are reservations or not
- When an item is returned late without having reservations, 4 days of suspension will be applied.

- When an object with reservations is returned late, 5 days of suspension will be applied.

Once all parameters have been filled in, click on the SUBMIT button to create a new profile with its own behaviour.

Always remember that we can see the profiles matrix as a list in which it can be verified which policy the created profile applies to each user and object category.

#### Application of a profile.

Once a profile has been edited clicking on SUBMIT will apply the profile on the new transactions. Remember that there should be loanobjects available for loaning and also a user category corresponding to the profile. This last issue is particularly important and often not easy to assure, esp. when users have been imported from other sources. Whenever EmpWeb doesn't recognize a user-category no transactions with that user will be possible ! Also when the user-category cannot be retrieved (check e.g the 'mapping' of the user-fields in the file `/[empweb]/dbws/WEB-INF/conf/isis_ABCD_usersconfig.xml`) an error message mentioning this condition will be shown.

## **III.2 Using the daily transactions**

### **III.2.1 Checking the availability of a book**

The EmpWeb installation package is pre-defined to do lending transactions with the MARC bibliographic database.

When you go to MySite (e.g. in a 'localhost' installation for testing : <http://localhost:9090/site>) and access the MARC database directly, if you do a successful search an Availability button will appear at the left of each item registered as a loan object. The association is done with the Control Number field as key.

Click the Availability button and a window opens displaying the available copies of the selected record.

Click the Availability button and the existing copies are shown.

As you can see, it is possible to make a reservation of this item clicking the Place Reservation button. This will open the MySite page to execute the reservation. The program will ask for login/password to access the Users database.

To have access to the Users (or patrons) database you must go to the main ABCD page (localhost:9090/) and login as a user with Loans Administrator profile. In the database you can Edit/Create/Modify users applying the necessary permissions to access MySite.

The loanobjects database was included in this distro to be shown as one of the editable databases in ABCD for testing purposes only. Remember that loanobjects is an internal database administered by ABCD and is not supposed to be directly edited by the final user.

By editing the records you can load data in the different fields. Field 001 is the ID-unique identifier for the record; field 10 is the database name (MARC in our case) and the repeatable field 959 contains the unique identifier for each

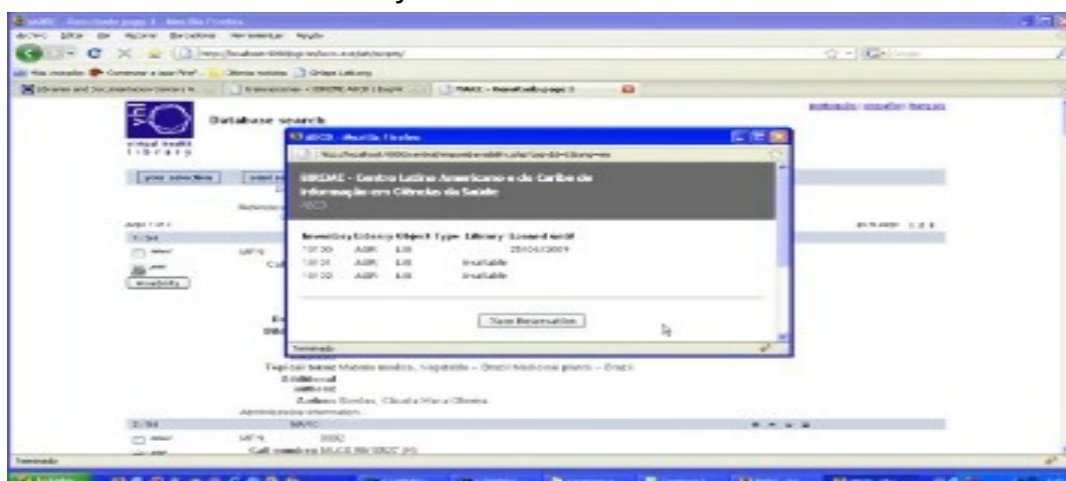
loanable copy.

### III.2.2 Reservations with EmpWeb.

Reservations can be made from 2 access points : MySite or as another transaction from the operator's EmpWeb interface.

Consulting the availability of items for a selected loanobject, one can reserve it from MySite; the system will allow to make the reservation when the requested title does not have copies available and in addition the user complies with the user-profile defined conditions (e.g. not being suspended).

When making a reservation if copies are available, the system does not allow to make this reservation but will indicate that an available copy has to be collected from the library.



When we go to MySite we can see that the status now indicates that the reservation has been done today and which title has been reserved.

Now we can record a return transaction of an item with reservation, this information will be reflected immediately in the waiting list of reservations of the principal EmpWeb window

When we try to realize a loan on this item which has a reservation by another user, EmpWeb informs us that the number of running loans do not suffice for the confirmed reservations in order to allow this transaction. So an reserved item will not be issued on loan.

Automatically in the MySite page of the user the availability of the reserved item will be reflected and also it will give information on the number of days it



will remain reserved to be collected. This number of days to be available for collection by the user will be taken from the EmpWeb policy.

At the time of creating a loans transaction or canceling a reservation, the information in the main EmpWeb page will be updated with the actual status of the transactions.

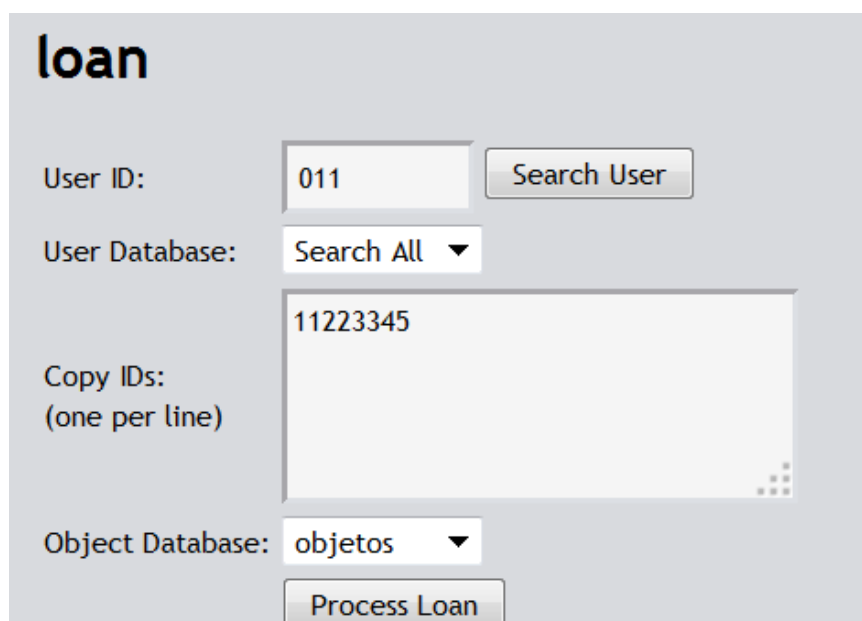
### III.2.3 Issuing a book

Putting a book on loan, or issuing it, is of course the most basic operation of any loans-system. The idea is to keep this as simple as possible in view of the potentially (very) high 'traffic' or number of such transactions to be performed every day, often also simultaneously from several work-posts or issuing desks in librerie branches.

Since EmpWeb avoids the overhead of a 'document oriented' database like ISIS but simply uses the matrix-based model of a relational database with just a few tables, such performance can be considered to be quite high.

The actual ergonomics are kept to a minimum :

- identifying the object, preferably with a barcode-scanner
- identifying the user, again preferably with a barcode-scanner
- and clicking on 'process loan'



The screenshot shows a web form titled 'loan' on a light gray background. The form contains the following elements:

- User ID:** A text input field containing '011' and a 'Search User' button to its right.
- User Database:** A dropdown menu showing 'Search All' with a downward arrow.
- Copy IDs:** A label 'Copy IDs: (one per line)' to the left of a large text area containing the number '11223345'. The text area has a small grid of dots in the bottom right corner.
- Object Database:** A dropdown menu showing 'objetos' with a downward arrow.
- Process Loan:** A button located at the bottom center of the form.

will trigger the whole pipe-line of loan-decisions and when all came out positive the loan transaction will be finished with positive result 'successfully'. That is all... just some extra information with links being provided, e.g. to the transaction record itself and to possible actions like printing or reviewing the user status.

## loan

### Loan Result

Transaction sucessful for:

- [11223345](#)

### Transaction result details

Success processing transaction for ID:11223345.

### Loan Information

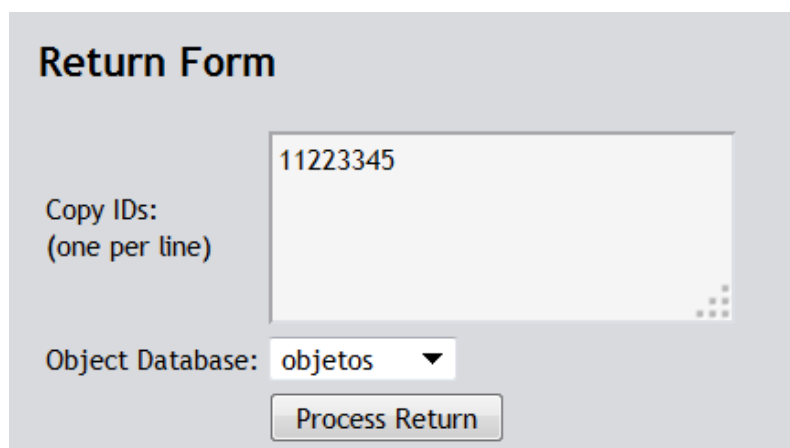
Transaction Id: [20141113\\_10FCA0](#)  
User Id: [01\(isis\)](#)  
Copy Id: [11223345\(objetos\)](#)  
Profile: [20141105\\_0E6E20](#)  
Date: 11/13/14 7:19 PM  
End date: 11/18/14 3:00 PM  
Location: AGR  
Operator: admin

### Actions

- [Print](#)
- [View user status](#)

## III.2.4 Returning a book

Compared with the issuing of an object, returning an object is even simpler : only the object has to be identified and the resulting information includes some links :



The image shows a web form titled "Return Form". It has a light gray background. On the left, there is a label "Copy IDs: (one per line)". To its right is a large text input field containing the number "11223345". Below the input field, there is a label "Object Database:" followed by a dropdown menu showing "objetos" with a downward arrow. At the bottom center, there is a button labeled "Process Return".

## Return Result

Transaction successful for:

- [11223345](#)

### Transaction result details

Success processing transaction for ID:11223345.

## Return Information

Transaction Id: [20141113\\_110F90](#)  
User Id: [01\(isis\)](#)  
Copy Id: [11223345\(objetos\)](#)  
Record Id: [1\(objetos\)](#)  
Loan Id: [20141113\\_10FCA0](#)  
Profile: [20141105\\_0E6E20](#)  
Loan date: 11/13/14 7:19 PM  
Return date: 11/13/14 7:24 PM  
Location: AGR  
Operator: admin

### Actions

- [Print](#)

### III.2.5 Renewing a loan-transaction

The renewal of a loan involves checking quite some conditions, e.g. whether reservations have been created on one of the copies of the title and still another copy for that reservation is available, as well as the usual conditions of the user-status.

The renewal interface simply asks for the ID of the object (the barcode of the book e.g.) and will then display the result, which – if positive – looks like this :

## renewal

### Renewal Result

Transaction successful

### Loan Information

Transaction Id: [20141113\\_112740](#)  
User Id: [01\(isis\)](#)  
Copy Id: [11223345\(objetos\)](#)  
Profile: [20141105\\_0E6E20](#)  
Date: 11/13/14 7:31 PM  
End date: 11/18/14 3:00 PM  
Location: AGR  
Operator: admin

## III.2.6 Reservation

Books, or more generally 'objects' can be reserved not only by the end-user from the OPAC (see above) but also by the librarians from the EmpWeb interface itself, by definition obeying the same rules of course.

The interface first asks whether a reservation is to be created or to be canceled.

In case of a reservation creation, the book first needs to be 'identified' by the librarian, with some data either on the object ID, the Control Number of the catalog, the title or the author.

### Object Query

Record Id:	<input type="text"/>
Copy Id:	<input type="text" value="11223345"/>
Title:	<input type="text"/>
Author:	<input type="text"/>
Object Database:	<input type="text" value="objetos"/>
<input type="button" value="Search Object"/> <input type="button" value="Reset Form"/>	

After the identification the librarian has to click on the control number of the object and the next available copy will be put aside, followed by the identification of the user who wants to reserve :

In case the reservation for this specific user is successful, a message will confirm it. But if for some reason, like in the example below, a copy of the title is still available, the librarian will be notified as such on this status and an instruction to pick up the available copy will be given :

## reservation

### Reservation Result

Transaction failed for:

ID	Problem
<input type="checkbox"/> 1	There are items available for this publication. Pick up from the library
<input type="button" value="Retry the selected transactions"/>	

### Transaction result details

Error processing transaction for ID:1.

There are items available for this publication. Pick up from the library  
Process name: checkAvaibility

### III.2.7 Dealing with fines and suspensions

In function of the rules defined in the policy/profile, in certain cases like overdue loans, users can be penalized with a fine or even be suspended for a defined span of time.

This process is either mechanically executed by the system itself (e.g. in case of a late return) or manually by the librarian. In this 2<sup>nd</sup> case the librarian can also add 'observations' to explain the reason or conditions.

## suspension

### Suspension Forms

Choose the desired task

- [Create suspension](#)
- [Cancel suspension](#)

**suspension**

User ID: 011

User Database: isis

Days suspended: 1

Obs: This is an example suspension for ISIS-user 011.

Suspensions and fines have their own tables in the transa-database which EmpWeb maintains in the SQL-environment.

## Create suspension

### Create Suspension Result

Transaction successful

### Issued suspension

Transaction Id: [20141113\\_115AC0](#)  
User Id: [011\(isis\)](#)  
Suspension creation date: 11/13/14 7:44 PM  
Suspension type: M  
Days suspended: 1  
Suspended from: 11/17/14 6:46:36 PM  
Suspended until: 11/18/14 6:46:36 PM  
Obs: This is an example suspension for ISIS-user 011.  
Location: AGR  
Operator: admin

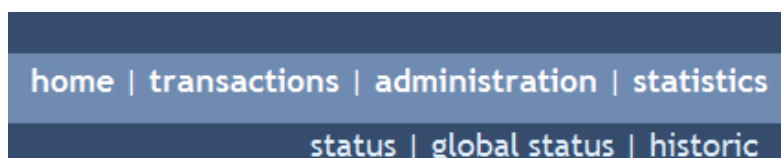
#### Actions

- [Print](#)
- [View user status](#)

## III.2.8 Reports and statistics

The last part of the discussion of the use of EmpWeb by the librarian is the reporting/statistics module. Differently from the ABCD Central Loans reports/statistics, only pre-defined reports and

statistics can be requested for, but then either for the whole network of libraries covered by EmpWeb or for an individual library or branch. In addition reports/statistics can be requested for the actual status or for a time-span ('historical').



The following screenshot shows the list of current statistics available for the 'current library' (AGR) :

## Current status

### Status Figures

- [AGR library figures](#)

### Status Reports: AGR

- [Current loans](#)
- [Overdue objects](#)
- [Active suspensions](#)
- [Pending fines](#)

When asking e.g. for the library figures for loans of the library 'AGR' something like the following table will be shown with a lot of details (and links) on the loans listed :

### List of Loans

#### List of Loans: AGR

[New search](#) | [Printable version](#)

page: 1

<a href="#">Loan Id</a>	<a href="#">Record title</a>	<a href="#">Copy Id</a>	<a href="#">Record Id</a>	<a href="#">Loan date</a>	<a href="#">Return date</a>	<a href="#">Location</a>	<a href="#">User Id</a>	<a href="#">User name</a>
<a href="#">20141113_112740</a>	Pobreza, cidadania e segurança /	11223345(objetos)	<a href="#">1(objetos)</a>	11/13/14 7:31:00 PM	11/18/14 3:00:00 PM	AGR	<a href="#">01(isis)</a>	Guilárte, Angela Margarita

page: 1

The next report option is on the 'Global Status' statistics :

## Current status

### Global Status Figures

- [All libraries figures](#)

### Global Status Reports

- [Current loans](#)
- [Overdue objects](#)
- [Active suspensions](#)
- [Pending fines](#)

Clicking on the 'All libraries figures' link will give an overview for all libraries :

#### All libraries stats

Library	Lent books	Overdue objects	Active suspensions	Pending fines
AGR	<a href="#">1</a>	<a href="#">0</a>	<a href="#">2</a>	<a href="#">0</a>
ARQ	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>
ING	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>
VET	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>	<a href="#">0</a>
total	1	0	2	0

If one clicks on the link for any more specific transaction type, the following type of screen is delivered, with a list of all active transactions of that type, in this case 'suspensions' :

#### List of Active Suspensions

##### List of Active Suspensions: AGR

[New search](#) | [Printable version](#)

page: 1

<a href="#">Suspension Id</a>	<a href="#">Date</a>	<a href="#">Duration</a>	<a href="#">Suspended until</a>	<a href="#">Obs</a>	<a href="#">User Id</a>	<a href="#">User name</a>
<a href="#">20141113_1080CQ</a>	11/13/14 6:46:36 PM	4	11/17/14 6:46:36 PM		<a href="#">011(isis)</a>	Egbert de Smet
<a href="#">20141113_115ACQ</a>	11/13/14 7:44:44 PM	1	11/18/14 6:46:36 PM	This is an example suspension for ISIS-user 011.	<a href="#">011(isis)</a>	Egbert de Smet

page: 1

Finally the statistics can also be created as 'historical listings' :



## Historic reports

### Historic reports

#### Loans

- [Historic loans by user](#)
- [Historic loans by record](#)
- [Historic loans by copy](#)

#### Fines

- [Historic fines by user](#)

#### Suspensions

- [Historic suspensions by user](#)

As a final example we illustrate what is retrieved when clicking on the historical list of loans while clicking on any of the resulting links will give more details of that specific item (object, transaction or user) :

### List of Loans

#### List of Loans: all libraries

[New search](#) | [Printable version](#)

page: 1

<a href="#">Loan Id</a>	<a href="#">Record title</a>	<a href="#">Copy Id</a>	<a href="#">Record Id</a>	<a href="#">Loan date</a>	<a href="#">Return date</a>	<a href="#">Location</a>	<a href="#">User Id</a>	<a href="#">User name</a>
 <a href="#">20141105_0E9DB0</a>	Pobreza, cidadania e segurança /	<a href="#">11223345(objetos)</a>	<a href="#">11(objetos)</a>	11/5/14 4:37:47 PM	11/10/14 3:00:00 PM	AGR	<a href="#">011(isis)</a>	Egbert de Smet
 <a href="#">20141105_0EA340</a>	Pobreza, cidadania e segurança /	<a href="#">11223345(objetos)</a>	<a href="#">11(objetos)</a>	11/5/14 4:39:16 PM	11/10/14 3:00:00 PM	AGR	<a href="#">011(isis)</a>	Egbert de Smet
 <a href="#">20141113_10FCA0</a>	Pobreza, cidadania e segurança /	<a href="#">11223345(objetos)</a>	<a href="#">11(objetos)</a>	11/13/14 7:19:38 PM	11/18/14 3:00:00 PM	AGR	<a href="#">01(isis)</a>	Guilárte, Angela Margarita
 <a href="#">20141113_112670</a>	Pobreza, cidadania e segurança /	<a href="#">11223345(objetos)</a>	<a href="#">11(objetos)</a>	11/13/14 7:30:47 PM	11/18/14 3:00:00 PM	AGR	<a href="#">01(isis)</a>	Guilárte, Angela Margarita
<a href="#">20141113_112740</a>	Pobreza, cidadania e segurança /	<a href="#">11223345(objetos)</a>	<a href="#">11(objetos)</a>	11/13/14 7:31:00 PM	11/18/14 3:00:00 PM	AGR	<a href="#">01(isis)</a>	Guilárte, Angela Margarita

page: 1

This concludes the overview of available statistics in EmpWeb, and indeed also the discussion of the use of the EmpWeb interface.

## IV. Annexes

### IV.1 EmpWeb Pipelines and Groovy.

#### IV.1.1 The concept of Pipelines and Groovy.

All operations realized by EmpWeb correspond to a pipeline.

We define a **pipeline** as a set of ordered processes, sequential or simultaneous, in which the output of one process is the input of the following one.

Groovy is a non-declarative programming language, with a syntax very similar to JAVA and with lots of programming facilities. The specifications of the Groovy language can be consulted at: <http://groovy.codehaus.org/>

The pipelines with which EmpWeb works in this distribution package of EmpWeb can be divided in the following groups :

Global pipelines. (Configuration Pipelines List)

### Transaction Pipelines Administration

#### Configuration Pipelines List

Pipeline Name	Configuration File	Pipeline Actions
conf-getLibraries	C:\ABCD\empweb\engine\WEB-INF\conf\conf-pipes\conf-getLibraries.xml	<a href="#">Editor</a>
globalenvironment	C:\ABCD\empweb\engine\WEB-INF\conf\conf-pipes\globalenvironment.xml	<a href="#">Editor</a>

#### Transaction Pipelines List

These pipelines define values used both in the graphical interface and in other pipelines. Here we find global values like e.g.

- The time-tables of opening hours of each library,
- the type of material registered for per-hour loans,
- the maximum time allowed for per hour loans,
- etc.

Transaction Pipelines (Transaction Pipelines List)

## Transaction Pipelines List

Pipeline Name	Configuration File	Pipeline Actions
cancelwait	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\cancelwait.xml	<a href="#">Editor</a>
cancelfine	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\cancelfine.xml	<a href="#">Editor</a>
suspension	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\suspension.xml	<a href="#">Editor</a>
fine	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\fine.xml	<a href="#">Editor</a>
renewal	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\renewal.xml	<a href="#">Editor</a>
loan	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\loan.xml	<a href="#">Editor</a>
cancelsuspension	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\cancelsuspension.xml	<a href="#">Editor</a>
return	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\return.xml	<a href="#">Editor</a>
wait	C:\ABCD\empweb\engine\WEB-INF\conf\trans-pipes\wait.xml	<a href="#">Editor</a>

These pipelines define the transactions which the main server of EmpWeb manages and of which the pipelines will be called by the corresponding processes of the graphical interface.

## Reports pipelines. (Statistics Pipelines List)

### Statistics Pipelines List

Pipeline Name	Configuration File	Pipeline Actions
stat-status-loans	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-loans.xml	<a href="#">Editor</a>
stat-trans-by-ids	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-trans-by-ids.xml	<a href="#">Editor</a>
stat-status-fines	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-fines.xml	<a href="#">Editor</a>
stat-status-lateLoans	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-lateLoans.xml	<a href="#">Editor</a>
stat-status-suspensions	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-suspensions.xml	<a href="#">Editor</a>
stat-record-availability	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-record-availability.xml	<a href="#">Editor</a>
stat-hist-user	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-hist-user.xml	<a href="#">Editor</a>
stat-hist-loans	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-hist-loans.xml	<a href="#">Editor</a>
stat-hist-fines	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-hist-fines.xml	<a href="#">Editor</a>
stat-status-waits-assigned	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-waits-assigned.xml	<a href="#">Editor</a>
stat-status-waits	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-waits.xml	<a href="#">Editor</a>
stat-status-counts	C:\ABCD\empweb\engine\WEB-INF\conf\stat-pipes\stat-status-counts.xml	<a href="#">Editor</a>

These pipelines are used at the moment of presenting a report. Such report will give an account of the requested information where this can be presented in the formats defined by the processes of the pipeline.

### IV.1.2 Detailed discussion of a pipeline.

A pipeline is a set of processes, which are executed in a consecutive or simultaneous way, ordered and sharing data. Let us consider some of these :

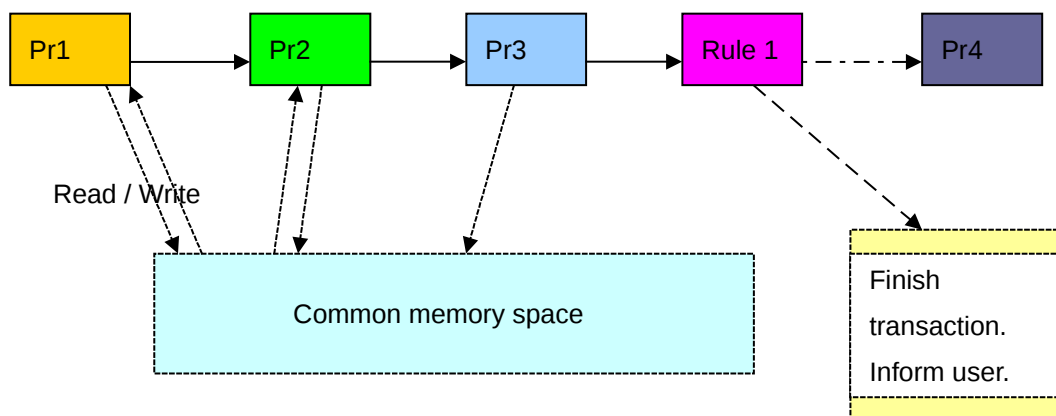
## The ABCD EMPWEB manual

Enabled	Type	Nombre	Descripción	Acciones
<input checked="" type="checkbox"/>	rule	GetUser	Get User DOM from (userId, userDb)	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	ExtractUserClass	Extract the user class from the user XML and store it in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetObject	Get Object DOM (mods) from (copyId/recordId, objectDb)	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	ExtractObjectCategory	Extract the object category from the object XML and store it in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetProfile	Gets a Profile for the userClass and objectCategory stored in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	AdjustProfileValues	Adjusts some of the Profile's values to the calculated ones.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	PublishTimestampAdjustments	Publica al TC horas de devolución, de expiración de reserva, de inicio de reserva, y excepciones	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	Lock	Logical lock of UserStatus and ObjectStatus	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetUserStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetObjectStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetExistingWaits	Finds an existing Reservation for the user that matches the object we are lending.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	PucvObjetoEsDeBiblioteca	Verifica si el objeto pertenece a la biblioteca donde se extra realizando la transaccion.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	objectAlreadyLent	Checks whether the object is already lent.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	CheckValidityDateIsNotNull	Check null values in User cards	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	HasFineOrSuspension	Verifies if the user has fine or suspension.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	LoanIfLate	Can we loan to a late user?	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	UserQuantities		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	checkAvailability	Check if this loan is not interfering the reservation queue	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process(Script)	obtainLibrariesInformation		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	CreateLoan	Creates a Loan object in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	ValidateAvailability	Validates availability to making the loan	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>

Since based on the pipeline concept the entry of a process is the output of a previous process, exceptions can occur: e.g. This will be the case with the first entry which could be based on data entered by the user. In the case of EmpWeb, an 'space' of shared objects is used from which the processes gather and request their values to shape the transaction as such.

There are two types of processes in EmpWeb, **rules** y **proper processes**.

1. The rules use the “common space for information exchange” to check specific values and the rule can return, in function of these values and the negotiated rule, “true” o “false” as a result, indicating with this result if the pipeline should continue or stop.
2. The proper processes interact with the common space of memory to store values with two possible objectives : either to finish the complete transaction or to hand over values to subsequent rules in order to test negotiating rules.



**1. Objects accessible during transaction realisation. Common memory space.**

It is important to underline that in the common memory space it is possible to get and store information which defines a couple (identification, value). This way the identifiers were previously defined and it is based upon these couples (identification, value) that the rules and processes can interact.

One of the common memory spaces most used by EmpWeb is the one named **TransactionContext**. In the definition of EmpWeb this space can gather or store values at the moment of running a transaction.

The concept of TransactionContext corresponds to a hash, which puts a key and develops a value from this key. This value can be linear like a String or an Empweb Object.

The following table summarizes the keys and their meaning for storage in the TransactionContext.

Clave	Tipo de valor
LOCAL_ENV	Environment of local parameters for the process. List of values.
GLOBAL_ENV	Environment of global parameters for the process. List of values.
USER_DOM	DOM of the user. Object
OBJECT_DOM	DOM del objeto. Objeto.
USER_ID	ID of the current user. String.
USER_DB	Database of the current user. String.
COPY_ID	Identification of the current copy. String.
VOLUME_ID	Volume of the current copy. String.
RECORD_ID	Identification of the current record. String.
OBJECT_DB	Current database of loanobjects. String.
OPERATOR_ID	Identification of the operator who manages the transaction. String.
DESK_OR_WS	Only for reservations. Whether the transaction comes from EmpWeb or from mySite
OBJECT_LOCATION	Identification of the library of the current copy
OBS	Observations entered by the user in the form. Only applicable in transactions like suspension or fines.
FINE_AMOUNT	Current fine amount. Real value.
PAID_AMOUNT	Amount paid for a fine as entered by the operator. Real value.
DAYS_SUSPENDED	
USER_CLASS	User type on which the transaction will be run.
OBJECT_CATEGORY	Object type on which the transaction will be run.
USER_STATUS	
USER_CLASS	User type for the current transaction. String.
OBJECT_STATUS	
PROFILE	Contains the object of the current transaction profile class. Defines the user category and object type for the current transaction.

### IV.1.3 Detailed analysis of some basic classes of EmpWeb when running a transaction.

In quite some pipelines, we will find the following processes invoked as initial processes. It should be noted that these classes have been compiled in JAVA and it is not possible to modify them without recompiling the application.

The comments are taken from the javadoc documentation.

#### **GetUser**

Clase: net.kalio.empWeb.engine.rules.GetUser

## *The ABCD EMPWEB manual*

It gets the userCollection XML from the user database and does some validity checks.

The Process expects to find the `userId` and `userDb` in the `TransactionContext` under `TransactionContext.USER_ID` ("`userId`") and `TransactionContext.USER_DB` ("`userDb`") keys, respectively.

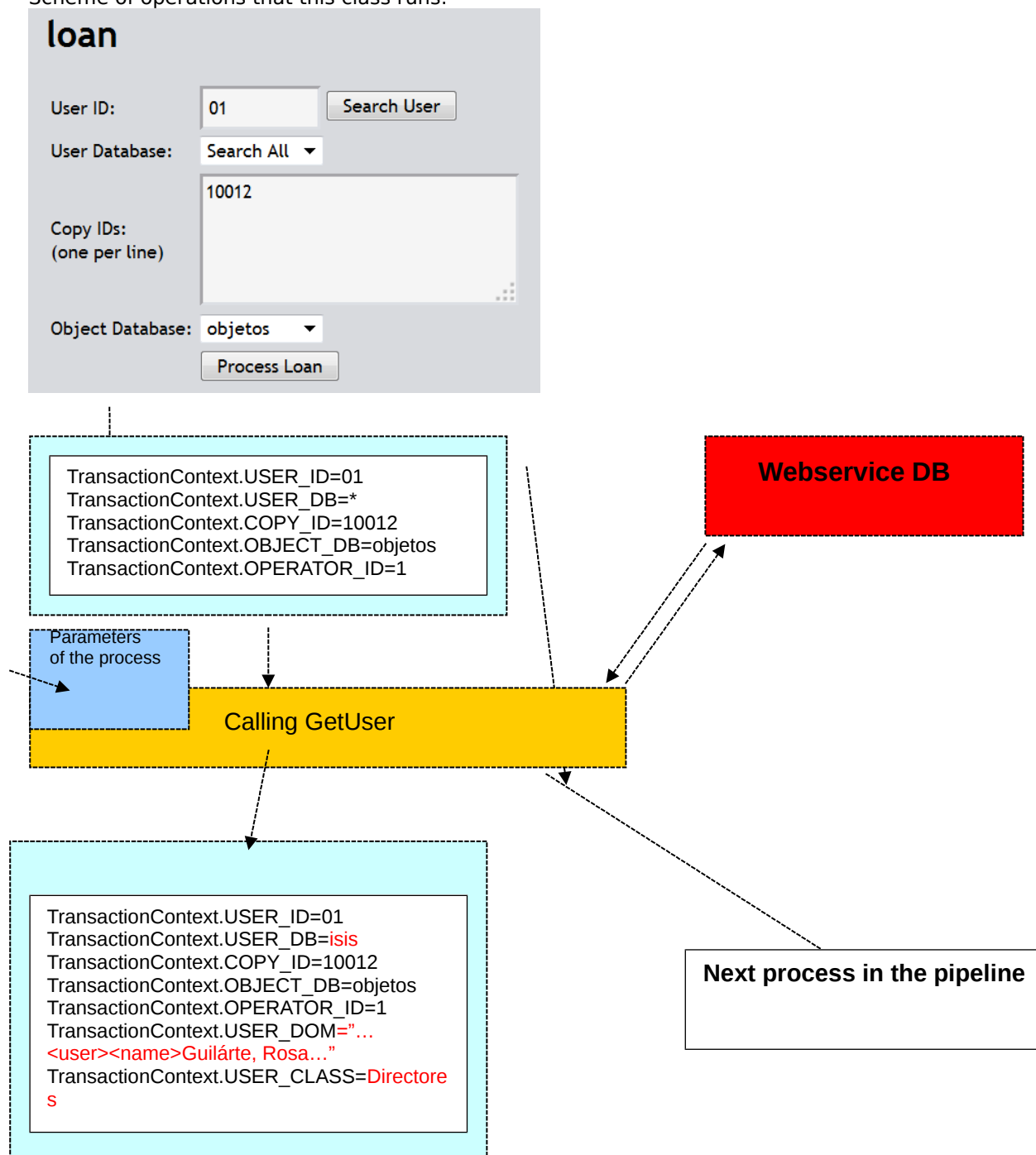
The `userCollection` DOM object will be stored in the `TransactionContext` with `setUserDOM`.

If the `userCollection` does not contain at least one user, the Process fails. If it contains more than one, usually only the first one will be used by the following Processes.

The Process accepts a boolean parameter `checkValidity` (default: `true`). If the parameter is "`true`" or missing then the user `expirationDate` and state (`active/inactive`) will be checked and the Process may fail with an appropriate `ProcessResult` message.

(For the meaning of `expirationDate` and state, read the <http://kalio.net/empWeb/schema/users/v1/schema>)

Scheme of operations that this class runs:



Description: It gets a MODS modsCollection from the database representing the object MODS .

The database is specified in the TransactionContext's objectDb.

The object may be specified by copyId or recordId, depending on the mode in which this class is used.

The mode is specified with a "mode" parameter in the params of the transaction pipeline XML. The value of "mode" can be "copyId" or "recordId".

For "copyId" (the default behavior), it performs a searchObjectsById on the objectDB web service. For "recordId" it will perform a searchObjects with a recordId element.

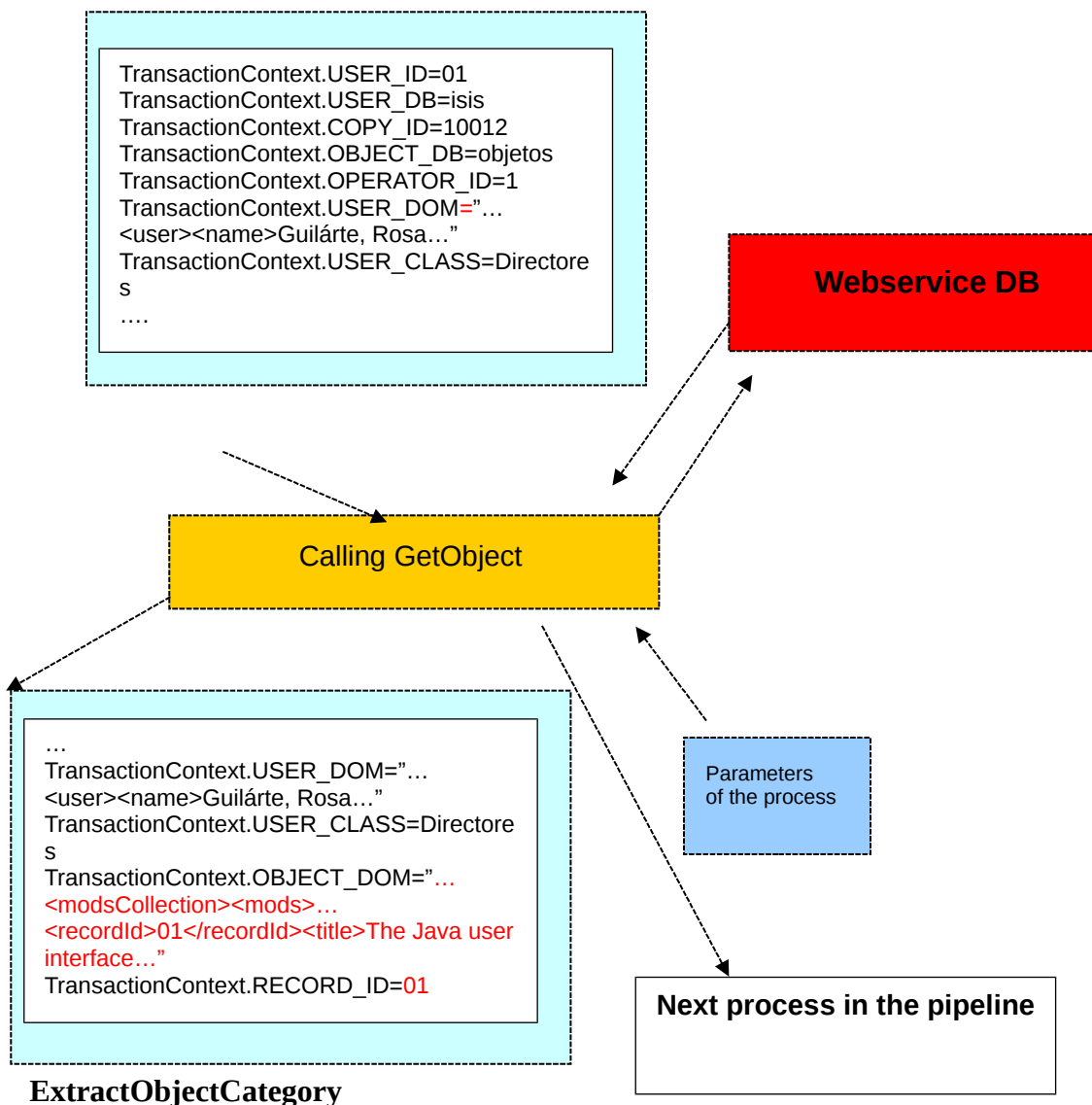


The MODS modsCollection DOM will be stored in the TransactionContext with setObjectDOM.

For convenience, in the case of "copyId" mode, the mods:recordInfo/mods:recordIdentifier will be stored in the TransactionContext under the "recordId" key.

The modsCollection will usually contain just one mods element. It MAY contain more than one but most transactions will use only the first mods. If modsCollection is empty, this Process will fail.

If the mode is "copyId", you can use the boolean extractExtraCopyInfo parameter (default: false) which will extract the copyLocation and volumeId (if exists) and store them under TransactionContext.OBJECT\_LOCATION and TransactionContext.VOLUME\_ID respectively (if they do not exist already in the TransactionContext).



Description: Extracts the objectCategory from the object's EmpWeb holdingsInfo and stores the value in the TransactionContext under the well-known key "objectCategory".

## *The ABCD EMPWEB manual*

This Process accepts three parameters:

- mode: It can have the value "copyId" or the value "recordId" (defaults to "copyId")
- useDefault: it's a boolean and defaults to false
- dontConsider: it's a list of object categories separated by comma or semicolon. These object categories should be ignored when extracting an object's objectCategory

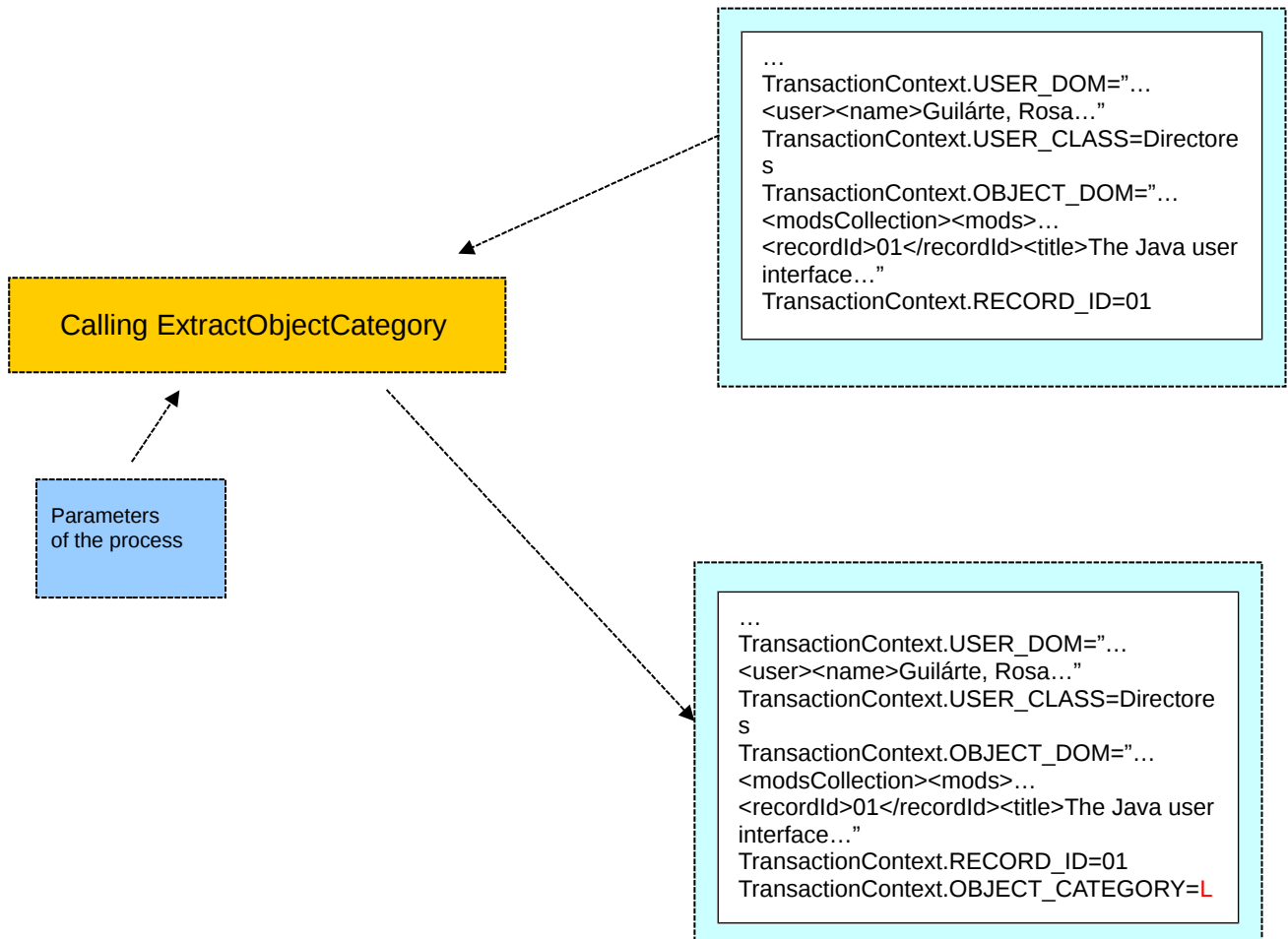
The algorithm works as follows: First, check whether the "mode" matches what is found in the TransactionContext. If the needed copyId or recordId isn't found, fail with an error message.

Then:

```
if (mode is recordId) then
  if (exists mods:mods for this recordId) then
    objectCategory:= "first objectCategory found in mods:mods that is not in the dontConsider
list"
  else
    Error: no_object_for_recordid
  end if
else // mode copyId by default
  objectCategory:= "the objectCategory of the given copyId"
end if

// if objectCategory still null and useDefault is false
if (objectCategory == null and not useDefault) then
  Error: record_with_no_object_category or copy_with_no_object_category
    according to mode
end if

store the objectCategory in the TransactionContext
```



As can be seen therefore, the processes run operations which store values in the TransactionContext, based on a particular key and the rules can test some of these values to implement the negotiation rule.

## **GetProfile**

Description: This Process obtains a Profile from the active Policy and stores it in the TransactionContext under the well-known name TransactionContext.PROFILE ("profile").

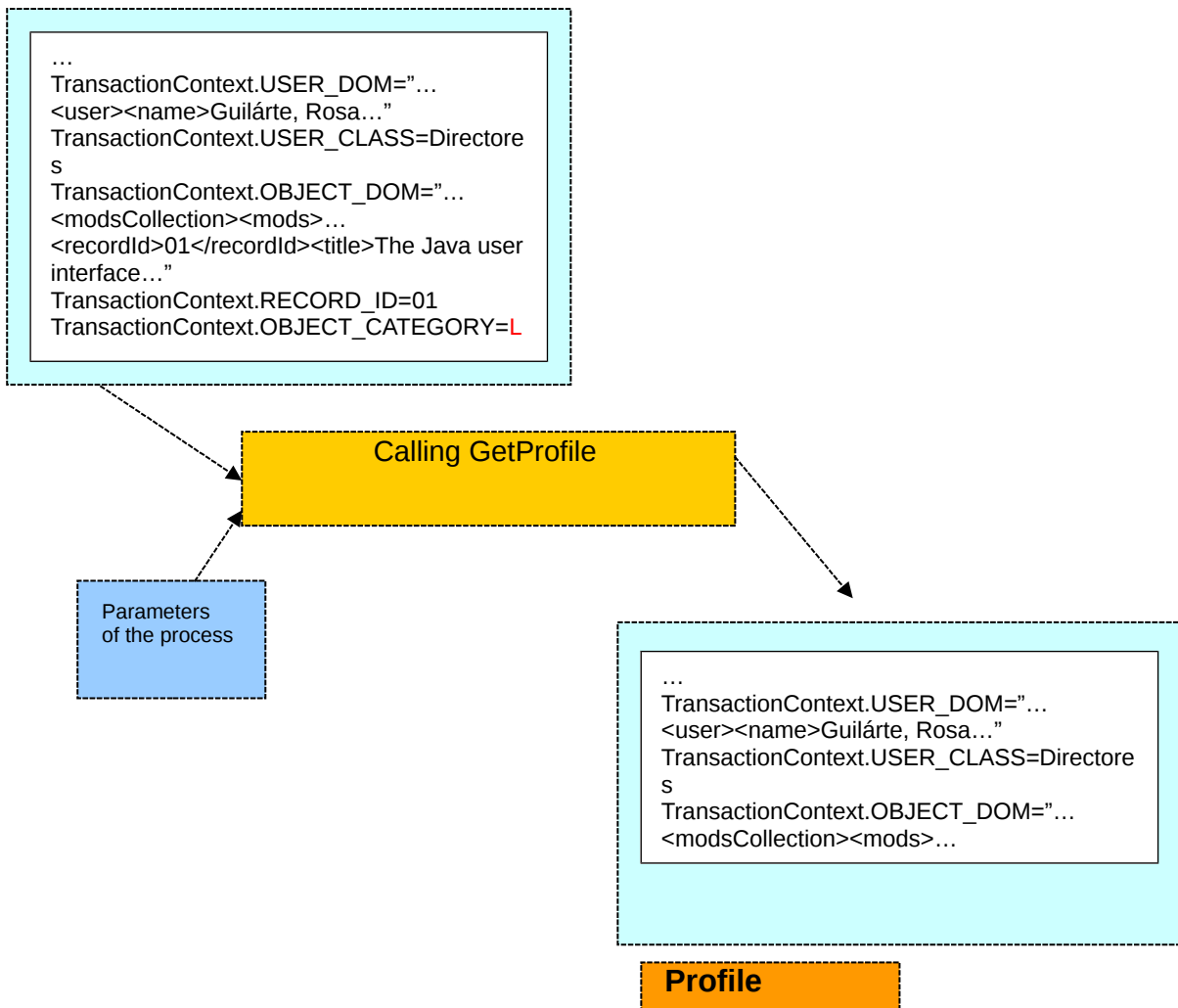
To select a Profile, it uses the "userClass" and "objectCategory" stored in the TransactionContext.

A "priority" parameter with valid values of "userClass" or "objectCategory" may be passed to the Process.

The behaviour is as follows:

- ✓ It attempts to get a Profile that matches (userClass, objectCategory)
- ✓ If such a Profile does not exist in the currently active Policy then:
  1. If the "priority" parameter is set to "userClass", attempt to get a Profile in this order:
    1. (TC.userClass, \*)
    2. (\*, TC.objectCategory)
    3. (\*, \*)
  2. If the "priority" parameter is set to "objectCategory", attempt to get a Profile in this order:
    1. (\*, TC.objectCategory)
    2. (TC.userClass, \*)
    3. (\*, \*)
  3. If the "priority" parameter does not exist, return an error
- ✓ If a Profile still couldn't be found, return an error

TC.userClass represents the userClass in the TransactionContext (and similarly for TC.objectCategory)



In the case of `GetProfile` we can see more clearly that in the `TransactionContext` objects are stored, which can later on be accessed by their properties or methods.

## **2. Object defined in EmpWeb.**

The following is a basic diagram with the objects which are accessible by Groovy scripts, together with their public access methods.

LoanImpl
+populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getRecordId(): String +setRecordId(id: String) +getCopyId(): String +setCopyId(id: String) +getObjectDb(): String +setObjectDb(id: String) +getProfile(): Profile +setProfile(p: Profile) +getStartDate(): String +setStartDate(date: String) +getEndDate(): String +setEndDate(date: String) +getRenewId(): String +setRenewId(id: String) +getOrdinalRenewal(): String +setOrdinalRenewal(number: String) +getOrdinalRenewalFromDesk(): String +setOrdinalRenewalFromDesk(number: String) +getOrdinalRenewalFromWS(): String +setOrdinalRenewalFromWS(number: String) +getReservationId(): String +setReservationId(id: String) +getOperatorId(): String +setOperatorId(id: String) +getLocation(): String +setLocation(id: String) +main(args: String) +setTypeOfTransaction(myType: String) +getTypeOfTransaction(): String

ReturnImpl
+populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getRecordId(): String +setRecordId(id: String) +getCopyId(): String +setCopyId(id: String) +getObjectDb(): String +setObjectDb(id: String) +getLoanId(): String +setLoanId(id: String) +getProfile(): Profile +setProfile(p: Profile) +getLoanDate(): String +setLoanDate(date: String) +getReturnDate(): String +setReturnDate(date: String) +getOperatorId(): String +setOperatorId(id: String) +getLocation(): String +setLocation(id: String) +main(args: String) +setTypeOfTransaction(myType: String) +getTypeOfTransaction(): String

WaitImpl
+populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getRecordId(): String +setRecordId(id: String) +getVolumeId(): String +setVolumeId(id: String) +getObjectDb(): String +setObjectDb(id: String) +getDate(): String +setDate(date: String) +getCancelDate(): String +setCancelDate(date: String) +getExpirationDate(): String +setExpirationDate(date: String) +getConfirmedDate(): String +setConfirmedDate(date: String) +getCancelId(): String +setCancelId(id: String) +getObs(): String +setObs(obs: String) +getProfile(): Profile +setProfile(p: Profile) +getOperatorId(): String +setOperatorId(id: String) +getLocation(): String +setLocation(id: String) +main(args: String) +setTypeOfTransaction(myType: String) +getTypeOfTransaction(): String

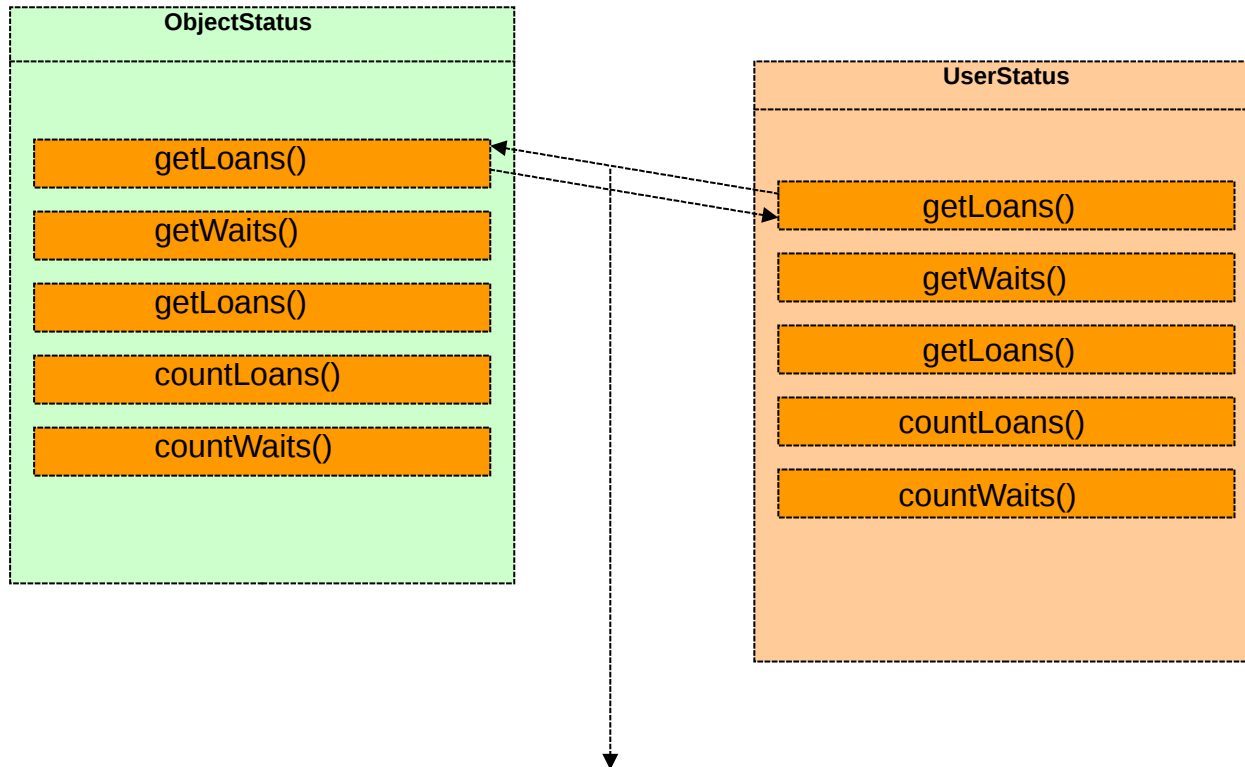
FineImpl
<pre> +populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getTableName(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getDate(): String +setDate(date: String) +getType(): String +setType(t: String) +getObs(): String +setObs(obs: String) +getAmount(): String +setAmount(amount: String) +getLocation(): String +setLocation(id: String) +getOperatorId(): String +setOperatorId(id: String) +getObject_copyId(): String +setObject_copyId(id: String) +getObject_objectDb(): String +setObject_objectDb(id: String) +getObject_recordId(): String +setObject_recordId(id: String) +getObject_profile(): Profile +setObject_profile(p: Profile) +getObject_loanStartDate(): String +setObject_loanStartDate(date: String) +getObject_loanEndDate(): String +setObject_loanEndDate(date: String) +getObject_daysOverdue(): int +setObject_daysOverdue(days: int) +getPaid_date(): String +setPaid_date(date: String) +getPaid_amount(): String +setPaid_amount(amount: String) +getRefId(): String +setRefId(id: String) +main(args: String) +setTypeOfTransaction(myType: String) </pre>

ProfileImpl
<pre> +populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +toXMLString(): String +toDOMElement(): Element +getId(): String +setId(id: String) +getUserClass(): String +setUserClass(uclass: String) +getObjectCategory(): String +setObjectCategory(ocategory: String) +getPolicyId(): String +setPolicyId(id: String) +getTimestamp(): String +setTimestamp(ts: String) +getLimitNames(): String +getLimitMap(): Map +getLimit(limitName: String): String +getLimit(limitName: String, defaultValue: String): String +setLimit(limitName: String, limitValue: String) +main(args: String) </pre>

SuspensionImpl
<pre> +populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getDate(): String +setDate(date: String) +getStartDate(): String +setStartDate(date: String) +getDaysSuspended(): int +setDaysSuspended(days: int) +getEndDate(): String +setEndDate(date: String) +getType(): String +setType(type: String) +getObs(): String +setObs(obs: String) +getCancelId(): String +setCancelId(id: String) +getLocation(): String +setLocation(id: String) +getOperatorId(): String +setOperatorId(id: String) +getObject_copyId(): String +setObject_copyId(id: String) +getObject_objectDb(): String +setObject_objectDb(id: String) +getObject_recordId(): String +setObject_recordId(id: String) +getObject_profile(): Profile +setObject_profile(p: Profile) +getObject_loanStartDate(): String +setObject_loanStartDate(date: String) +getObject_loanEndDate(): String +setObject_loanEndDate(date: String) +getObject_daysOverdue(): int +setObject_daysOverdue(days: int) +main(args: String) +setTypeOfTransaction(myType: String) +getTypeOfTransaction(): String </pre>



### 3. Special objects used in EmpWeb



Parallel methods. In the rules these will be frequently used, like e.g. to know or browse the list of loan and/or reservations which the user or objects has pending.

#### IV.1.4 Structure of a process or rule in a pipeline.

### Transaction Process Administration

#### Edit Transaction Process

##### Process Information

Pipeline Name	<a href="#">loan</a>
Nombre del Proceso	GetUser
Type	rule
Class	net.kalio.empweb.engine.rules.GetUser
Bundle	

##### Process Documentation

```
<doc>Get User DOM from {userId, userDb}</doc>
```

##### Process limits

##### Process parameters

```
<params>
  <!-- checks for expired or disabled user -->
  <param name="checkValidity">true</param>
</params>
```

As can be observed, the processes or rules have a first entry-box dedicated for the documentation, a second one for the specifications for the limits and a third one in which the parameters of the process are specified. Let's take the relevant ones one by one.

#### Limits published by a process or rule.

When a process or rule is dependent of a particular profile, let's assume e.g. that we need a

process which sends an alert to the operator when the actual number of loans is superior to X.

X is a limit which can have a value set as **x1** for the profile (Students, Books) and a value **x2** for the profile (Directors, Books).

In this case, it is indicated that X is a limit which will be used within the process or rule with the correct value for the profile.

The syntax for putting values in the limits is as follows :

```
<limits>
  <limit name="name">default value</limit>
</limits>
```

Let's suppose now that a new limit is created by a particular rule or process. This will imply that the value will need to be filled in in all profiles which EmpWeb currently has registered for the active policy. In the case that the profile to be used does not have a value filled in for the correct profile, the default value will be used.

It is important to emphasize that before the use of any process or rule which uses limits, The basic class GetProfile should have been called, given that if not the default value will always be used.

## Parameters for a rule or process.

The parameters of a particular rule or process can have specific individual behaviours specified during the use of the same process or rule in different pipelines.

The syntax of the parameters is :

```
<params>
  <param name="attribute">value</param>
</params>
```

An example can be obtained from the basic class UpdateDb, which is used in several pipelines, and in which different values could be used for the parameters for each use.

Example :

```
<params>
  <param name="transactionKeys">paymentFine,pendingFine</param>
  <param name="ignoreTransactionNotFound">true</param>
  <param name="storeUserStatus">true</param>
  <param name="storeObjectStatus">>false</param>
</params>
```

## Rule or process types of Groovy scripts.

Scripted processes or rules are special rules or processes which do not need the file of the application to be compiled fully and which allow interaction with the EmpWeb objects like for implementing particular negotiation rules.

The structure of a scripted rule or process that can be used consists of :

**Process Documentation**

<doc>Checks whether the object is already lent.</doc>

**Process limits**

**Process parameters**

```
<params>
  <param name="script">
    <![CDATA[
      objectStatus = tc.get('objectStatus');
      copyId = tc.get('copyId');

      loansList = objectStatus.getLoans();
      for (loan in loansList)
      {
        if (loan.getCopyId() == copyId)
        {
          msg.addText('en', "This object is already lent to user ${loan.getUserId()}!");
          msg.addText('es', "Este objeto ya está prestado al usuario ${loan.getUserId()}!");
          return false;
        }
      }

      return true;
    ]]>
  </param>
</params>
```

Enviar

ABCD 0.9  
2009 BIREME - Centro Latino Americano e do Caribe de Informação em Ciências da Saúde  
<http://www.bireme.br>

BIREME • OPAS • OMS

The same files are taken into account as with a basic rule or process of EmpWeb, only the Groovy script deals with the parameters.

The structure of this is similar to the one of any parameter of a rule or process.

```
<params>
<param name="script">
```

```
<![CDATA[
```

.... instructions of the script.

```
]]>  
</param>  
</params>
```

A Groovy script deals with a set of variables which we can call “magic variables” with which one can interact freely :

## First example of a Groovy script.


The first step for aggregating a Groovy script consists of creating a rule or process for the correct pipeline. Continuing the previous example, we will create a new scripted process in the Loans pipeline, which will send a message to the operator.

Clicking on the option Create Process the following window will appear :

### Confirmación

#### New Transaction Process

##### New Process Information

New process type	Rule (only checks, no side effects) 
New process name	ControlLoansByHour
New process class	net.kalio.empweb.engine.rules.GroovyInterpreter

Are you sure you want to create this process?

We select the name of a process or rule and indicate that the class which will interpret our script will be `net.kalio.empWeb.engine.rules.GroovyInterpreter`.

Once filled in this window, the process which we have just created will appear in the pipeline, inactive and at the end of the list.

Important therefore, since the pipeline is an ordered sequence, to put the process within the sequence of the pipeline at the right position.

This will imply, given that we pretend to create a validation rule, that the said rule executes before the registration of the transaction. Therefore, using the options 'Up/Down' allowing repositioning the process, one has to locate the process at the right position for its execution :

<input checked="" type="checkbox"/>	rule	GetUserStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetObjectStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	GetExistingWaits	Finds an existing Reservation for the user that matches the object we are lending.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	PucvObjetoEsDeBiblioteca	Verifica si el objeto pertenece a la biblioteca donde se esta realizando la transaccion.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	objectAlreadyLent	Checks whether the object is already lent.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	CheckValidityDateIsNotNull	Check null values in User cards	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	HasFineOrSuspension	Verifies if the user has fine or suspension.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	LoanIfLate	Can we loan to a late user?	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	UserQuantities		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	checkAvailability	Check if this loan is not interfering the reservation queue	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process(Script)	obtainLibrariesInformation		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	CreateLoan	Creates a Loan object in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	ValidateAvailability	Validates availability to making the loan	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	RemovePrevReservationFromStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	AddLoanToStatus		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input type="checkbox"/>	rule(Script)	ControlLoansByHour		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	UpdateDb		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	ReturnTransactionResults		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	finally	Unlock	Release the logic locks done at the beginning.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>

[Create new process](#)

Pressing repeatedly on the link Up we drop the new rule to be created before the execution of the Loan entity. So it stays in the position as can be seen under here :

<input checked="" type="checkbox"/>	rule	HasFineOrSuspension	Verifies if the user has fine or suspension.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	LoanIfLate	Can we loan to a late user?	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	UserQuantities		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule(Script)	checkAvailability	Check if this loan is not interfering the reservation queue	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process(Script)	obtainLibrariesInformation		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input type="checkbox"/>	rule(Script)	ControlLoansByHour		<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	process	CreateLoan	Creates a Loan object in the TransactionContext.	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>
<input checked="" type="checkbox"/>	rule	ValidateAvailability	Validates availability to making the loan	<a href="#">Up</a>   <a href="#">Down</a>   <a href="#">Editar</a>   <a href="#">Borrar</a>

Then we press the link Edit and we are ready to create our first Groovy script which validates a particular situation.

Each Groovy script will have a syntax mentioned earlier in its parameters :

### Process parameters

```
<params>
  <param name="script">
<![CDATA[
|

]]>
  </param>
</params>
```

**Warning :** If the rule is to be included in the process of basic execution, remember to activate it after edition, i.e. :

<input checked="" type="checkbox"/>	rule(Script)	LoanIfLate	Can we loan to
<input checked="" type="checkbox"/>	rule(Script)	UserQuantities	
<input checked="" type="checkbox"/>	rule(Script)	checkAvailability	Check if this l
<input checked="" type="checkbox"/>	process(Script)	obtainLibrariesInformation	
<input checked="" type="checkbox"/>	rule(Script)	ControlLoansByHour	
<input checked="" type="checkbox"/>	process	CreateLoan	Creates a Loan
<input checked="" type="checkbox"/>	rule	ValidateAvailability	Validates avai
<input checked="" type="checkbox"/>	rule	RemovePrevReservationFromStatus	

The rule or process has to be ticked for it to be executed.  
So now we can work on our first script.

The case to be evaluated, as we envisage, will be :

“...The users of type Coordinators will not be allowed to have more than 2 books at the same time per reading room or per hour ...”

If we put this in a pseudo-code acting on the Groovy objects, we will have :

Get the user type;

If the user type is **Coordinators** then

For (**loans by this user**)

If the loan is an object loanable per hour entonces

Count

End For

If **number counted** > 1 then

Send message

Return that the rule failed

End If

End If

Return that the rule accomplished.

In this step it will be important to document that in every Groovy script a set of “magic variables” can be used :



- Variable tc (TransactionContext)
- Variable params (the parameters of the current rule or process)
- Variable msg (messages which can be produced by the current script)

Therefore, reviewing the objects which we have available in the EmpWeb Objects universum, we can analyze how we can get the actual user's type.

If we look into the list of processes actually existent in the pipeline, we would see that there is already a process present which is named ExtractUserClass. This one fills in the value for **TransactionContext.USER\_CLASS**

#### Process and Rules

Enabled	Type	Nombre	Descripción	
<input checked="" type="checkbox"/>	rule	GetUser	Get User DOM from (userId, userDb)	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	ExtractUserClass	Extract the user class from the user XML and store it in the TransactionContext.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	GetObject	Get Object DOM (mods) from (copyId/recordId, objectDb)	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	ExtractObjectCategory	Extract the object category from the object XML and store it in the TransactionContext.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	GetProfile	Gets a Profile for the userClass and objectCategory stored in the TransactionContext.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	AdjustProfileValues	Adjusts some of the Profile's values to the calculated ones.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	process	PublishTimestampAdjustments	Publica al TC horas de devolucion, de expiracion de reserva, de inicio de reserva, y excepciones	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	Lock	Logical lock of UserStatus and ObjectStatus	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	GetUserStatus		<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	GetObjectStatus		<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule	GetExistingWaits	Finds an existing Reservation for the user that matches the object we are lending.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule(Script)	PucvObjetoEsDeBiblioteca	Verifica si el objeto pertenece a la biblioteca donde se esta realizando la transaccion.	<a href="#">Up</a>
<input checked="" type="checkbox"/>	rule(Script)	objectAlreadyLent	Checks whether the object is already lent.	<a href="#">Up</a>

Therefore we could start the Groovy script by checking this value :

```
tipousuario = tc.get(TransactionContext.USER_CLASS);
```

```
If (tipousuario=='Students')
```

```
{
```

```
}
```

```
return true;
```

With this we have implemented the first step, related to the check of the user type. Now we will check the current loans which the users has on his name.

For this, we will use the special object type indicated previously, in this case the UserStatus.

Using this, we can get a list of the objects which correspond to the loans of the actual user.

But how can we get the UserStatus in the current script ?

As in the previous case, when we check the actual pipeline, we will see that there is another process, already in the TransactionContext, giving the userStatus needed.

## Process and Rules

Enabled	Type	Nombre	Descripción	
<input checked="" type="checkbox"/>	rule	GetUser	Get User DOM from (userId, userDb)	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	ExtractUserClass	Extract the user class from the user XML and store it in the TransactionContext.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	GetObject	Get Object DOM (mods) from (copyId/recordId, objectDb)	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	ExtractObjectCategory	Extract the object category from the object XML and store it in the TransactionContext.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	GetProfile	Gets a Profile for the userClass and objectCategory stored in the TransactionContext.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	AdjustProfileValues	Adjusts some of the Profile's values to the calculated ones.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	process	PublishTimestampAdjustments	Publica al TC horas de devolucion, de expiracion de reserva, de inicio de reserva, y excepciones	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	Lock	Logical lock of UserStatus and ObjectStatus	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	GetUserStatus		<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	GetObjectStatus		<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule	GetExistingWaits	Finds an existing Reservation for the user that matches the object we are lending.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule(Script)	PucvObjetoEsDeBiblioteca	Verifica si el objeto pertenece a la biblioteca donde se esta realizando la transaccion.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule(Script)	objectAlreadyLent	Checks whether the object is already lent.	<a href="#">Uf</a>
<input checked="" type="checkbox"/>	rule(Script)	CheckValidityDatelsNotNull	Check null values in User cards	<a href="#">Uf</a>

So, we can access a key from the TransactionContext, which will return the userStatus of the current user.

Using this object userStatus, we can browse the loans of this user.

```
uStatus = tc.get(TransactionContext.USER_STATUS);
```

Which methods support userStatus? One of them is getLoans(), which will give us the loans of the current user :

```
loans = uStatus.getLoans();
```

We will now try to browse this list returned by the object userStatus and note the loans which are relevant to us.

We use a Groovy loop for this :

```
for (loan in loansList) { // Here we will analyze each individual loan}
```

If we analyze the object type Loan, we will find out that there is no method which gives us the loaned object type.

This is due to the fact that the Loan object has a reference to the profile (object type, user type) in which it was created.

LoanImpl
+populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +getId(): String +setId(id: String) +getUserId(): String +setUserId(id: String) +getUserDb(): String +setUserDb(id: String) +getRecordId(): String +setRecordId(id: String) +getCopyId(): String +setCopyId(id: String) +getObjectDb(): String +setObjectDb(id: String) +getProfile(): Profile +setProfile(p: Profile) +getStartDate(): String +setStartDate(date: String) +getEndDate(): String +setEndDate(date: String) +getRenewId(): String +setRenewId(id: String) +getOrdinalRenewal(): String +setOrdinalRenewal(number: String) +getOrdinalRenewalFromDesk(): String +setOrdinalRenewalFromDesk(number: String) +getOrdinalRenewalFromWS(): String +setOrdinalRenewalFromWS(number: String) +getReservationId(): String +setReservationId(id: String) +getOperatorId(): String +setOperatorId(id: String) +getLocation(): String +setLocation(id: String) +main(args: String) +setTypeOfTransaction(myType: String) +getTypeOfTransaction(): String

What we can do now is obtain the profile, from which the indeed can obtain the string which represents the loan object type:

ProfileImpl
~debug: boolean = false ~NS: String = "http://kalio.net/empweb/schema/profile/v1" ~NS_PREFIX: String = "prof" ~TABLE_NAME: String = "profiles" ~DEFAULT_HISTORIC: String = "false" ~PROFILE_ID_COL: String = "profile_id" ~USER_CLASS_COL: String = "user_class" ~OBJECT_CATEGORY_COL: String = "object_category" ~POLICY_ID_COL: String = "policy_id" ~HISTORIC_COL: String = Empweb15DB.HISTORIC_COL ~XML_COL: String = "xml" ~dom: Element ~jxdom: JXPathContext
<<create>>~ProfileImpl() <<create>>~ProfileImpl(e: Element) <<create>>~ProfileImpl(rs: ResultSet) +populate(e: Element) +getNamespace(): String +getNamespacePrefix(): String +toXMLString(): String +toDOMElement(): Element ~getClone(): ProfileImpl +getId(): String +setId(id: String) +getUserClass(): String +setUserClass(uclass: String) +getObjectCategory(): String +setObjectCategory(ocategory: String) +getPolicyId(): String +setPolicyId(id: String) +getTimestamp(): String +setTimestamp(ts: String) +getLimitNames(): String +getLimitMap(): Map +getLimit(limitName: String): String +getLimit(limitName: String, defaultValue: String): String +setLimit(limitName: String, limitValue: String) ~cleanupLimits() ~isHistoric(): boolean ~setHistoric(h: boolean) ~getMockupDOM(): Element ~getMockup(): Profile +main(args: String)

Thus, this part of the code in Groovy would be as follows :

```

Profile prof = loan.getProfile();
If (prof.getObjectCategory=='LBH')
{

    //count
}

```

Now the final version of our code would read :

```

<![CDATA[

usertype = tc.get(TransactionContext.USER_CLASS);
if (usertype == 'Coordinators')
{

```

```

uStat = tc.get(TransactionContext.USER_STATUS);
loans = uStat.getLoans();
counter = 0;
for (loan in loans)
{
    Profile prof = loan.getProfile();
    objCateg = prof.getObjectCategory();
    if (objCateg=='LBH')
    {
        counter++;
    }
}

if (counter>1)
{
    msg.addText("en_CL_UCV","It's impossible to loan to Coordinators more than 2
concurrent by hour books");
    msg.addText("es_CL_UCV","No es posible prestar mas de dos libros
concurrentes por hora a Coordinadores");
    return false;
}
}
return true;
]]>

```

If we dwell on the msg area, it displays a result of the transaction, and its syntax is as follows:

```
msg.add ("language identifier", "message");
```

Interestingly, in evaluating the entered script execution stopped; this is due to the fact that there is a loan per hour. This is because we have a loan for which the rule when applied does not meet the originally specified rule.

We can now conclude that the logics of the pseudo-cod was not correct, and we propose the correct version in what follows :

Get the user type;

Get the object type which the user wants to take;

If the user type is **Coordinators** AND **the object type is for reading room only** then

For ( **loans by this user**)

    If **the loan is of object type per hour** hen

        Count

    End For

    If **number counted** > 1 then

        Produce message

        Return rule failed

```
End If
End If
```

Return rule accomplished.

In the implementation we see that it comes out rather simple to assemble this operation :

```
<![CDATA[

UserType = tc.get(TransactionContext.USER_CLASS);
ActualObjectType = tc.get(TransactionContext.OBJECT_CATEGORY);

if (UserType=='Coordinators' && ActualObjectType =='LBH' )
{
    uStat = tc.get(TransactionContext.USER_STATUS);
    loans = uStat.getLoans();

    counter = 0;
    for (loan in loans)
    {
        Profile prof = loan.getProfile();
        objCateg = prof.getObjectCategory();
        if (objCateg=='LBH')
        {
            counter++;
        }
    }

    if (counter > 1)
    {
        msg.addText("en_CL_UCV","It is impossible to loan to Students more than 2
concurrent by hour books");
        msg.addText("es_CL_UCV","No es posible prestar mas de dos libros
concurrentes por hora a estudiantes");
        return false;
    }
}

return true;
```

When we execute this we can observe, for example with the user 01 who is a Coordinator and how has the item 10012 on loan, the followin screens :

## loan

User ID:

User Database:  ▼

Copy IDs:  
(one per line)

Object Database:  ▼

## loan

### Loan Result

Transaction failed for:

ID	Problem
<input type="checkbox"/> <a href="#">10013</a>	It's impossible to loan to Students more than 2 concurrent by hour books
<input type="button" value="Retry the selected transactions"/>	

### Transaction result details

Error processing transaction for ID:10013.

With this we can see clearly the usefulness of associating each message with the identifier for the corresponding language.

**Attention :** when sending messages to the MySite these ones would have to be repeated with the identifier en and es (or pt, fr, etc) but without the identifier part xx\_CL\_UCV.