



Caravel

# Database and entities creation process



© Copyright 2006 TransTOOLS, S.A. All rights reserved.

Information contained in this document is subject to changes without prior notice. These changes will be incorporated in new editions of the document.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc., in the United States, other countries or both.

Every company name, product name, or service name may be trademarks or service marks of their respective owners.

Document: database loading procedure 20040525 v1.1 en.doc

Version: 1.1

Date: May. 25, 2004

TransTOOLS, S.A.

<http://www.transtools.com>

---

## Contents

<b>CHAPTER 1. INTRODUCTION .....</b>	<b>4</b>
Folder <i>ant</i> .....	4
Folder <i>data</i> .....	4
<b>CHAPTER 2. CONFIGURATION .....</b>	<b>5</b>
<b>CHAPTER 3. ENTITYDATALOADER TASK .....</b>	<b>6</b>
Necessary files.....	6
How this task works.....	7
<b>CHAPTER 4. CREATECATALOGS TASK .....</b>	<b>8</b>
<b>CHAPTER 5. LOADDATA TASK.....</b>	<b>9</b>
Necessary files.....	9
How this task works.....	10
<b>CHAPTER 6. EXECUTION.....</b>	<b>11</b>
After execution.....	11

---

## Chapter 1. Introduction

The files at compressed will be used only to create and load databases, so they must be uncompressed in a different folder where the application will be installed.

---

### Folder *ant*

It contains the XML files that Ant needs to create and load the entities database and the application database. It also contains jar files with caravel classes and JDBC driver.

- *entityDataLoader.xml*. File of the task for the creation of entities database.
- *CreateCatalog.xml*. File of the task for the creation of application database.
- *LoadData.xml*. File of the task for loading tables of application database.
- *CleanDatabase.xml*. File of the task for deleting tables of application database. To use only if needed.
- *path.properties*. File with path information used by the XML files.
- Folder *cartools/utills*. This folder contains jar files: *caravel.rt.jar*, *caravel.lib.jar*, *caravel.resources.web.zip*, and, if needed, jar file with JDBC driver for the database manager.

---

### Folder *data*

It contains XML files with data to be loaded at entities database and application database. It also contains Caravel properties files.

- *caravel.xml.properties*. Client configuration file for Caravel.
- Folder *systemname*. It contains Server configuration file (*system.xml.properties*). It has the information to access to entities database and application database. It must be configured. The name of this folder must be the same than the value for property *name* at section *system* in file *caravel.xml.properties*. (E.g. *demosys*).
- Other folders. It contains data to be loaded in the entities database and application database.

---

## Chapter 2. Configuration

First of all create the databases in DB2 for entities and data, as explained at document *Database Creation Procedure DB2*.

Once you've created database, *system.xml.properties* (in folder *data/systemname*) have to be changed in order to specify the way to access to the created databases.

These properties have to be changed in section *system*, where it's specified how to access to entities database and section *jdbc*, where it's specified how to access to application database:

- user: a user that can access to entities database.
- password: password for the user.
- url: Specify the URL according the format for JDBC driver used.

Also, at section *system*, property *homeDir*, must be set with the value of a path where system will generate their resources. It'll be a path where system Manager process will run (now in the same machine where the application will run).

At section for the application configuration, properties *updatablePath* and *tapeDirectory* must be set with a path where application will generate some needed files, and the path for tape commands. A path in the machine where application will be run.

---

## Chapter 3. entityDataLoader task

This task creates tables and load the information in the entities database created previously.

Target *entityLoader* from the task needs to define:

```
<entityDataLoader regenerate="true" loadpath="{datadir}"
properties="{datadir}/caramel.db.xml.properties"
user="admin" password="admin" errorsDir="{path.errorres}">
</entityDataLoader>
```

The elements specified are:

- **loadpath**: name of the directory that contains folders with data to be loaded.
- **properties**: client properties file.
- **user and password**: administrator user to create database, they must have the same values than the properties *adminUser* and *adminPassword* at *system.xml.properties* file.
- **errorsDir**: folder where the status file for process will be created.

---

### Necessary files

Into the folder specified at element *loadpath* must be these folders:

- **Qsys**. It contains xml format files with the information for the system.
  - Files: *counters.xml*, *libs.xml*, *printers.xml*, *printers\_prtdev.xml*, *ptrdevs.xml*, *sysattr.xml*, *users.xml*, *userspaces.xml*, to load the tables with its same name.

For every library in *libs.xml* (system and application libraries), there must be a directory that it'll contain these folders:

- **dtaqs**. It contains XML files with the definition of the library data queues.
- **jobds**. Definition of the library job descriptions, a XML file for every one.
- **jobqs**. Definition of the library job queues, a XML file for every one.
- **msgfs**. It contains XML files with the content for the library message files.
- **msgqs**. Definition of the library message queues, a XML file for every one.
- **outqs**. It contains XML files with the definition of the library out queues.

---

## How this task works

First of all, all entities tables are created, then the content of the XML files at folder *qsys* is loaded into the entities.

For every library, are registered the different items that contains (outqs, dtaqs, jobds, etc.) in the entities database: inserting a record for every item in the entity.

For every message file, a new table is created and the content of the XML file at *msgfs* is loaded.

For every data queue, a new table is created and the content of the XML file at *datqs* is loaded.

---

## Chapter 4. CreateCatalogs task

This task creates some special tables into the database application. These tables are necessary to Caravel knows the information about physical and logical files. Task only creates tables, at LoadData task, for every item created, information in Caravel catalog will be inserted.

Tables for Caravel catalog are: sysdf, sysdaara, syspfmbr, syslfmbr, sysmodel and syspftrigger.

To know more about these tables, see document *Entity DB Structure*.

---

## Chapter 5. LoadData task

This task creates database tables for every file in the libraries to be loaded.

At target *loadData* in the task for every library to be loaded it must be specified this line:

```
<loadData properties="${datadir}/caravel.xml.properties"  
    user="admin" password="admin" baseDirectory="${basedir}/${datadir}  
    library="MYLIBRARY" errorsDir="${path.errorres}/ MYLIBRARY "  
"/>
```

The elements specified are:

- **properties:** client properties file.
- **user and password:** administrator user to create database, they must have the same values than the properties *adminUser* and *adminPassword* at *system.xml.properties* file.
- **baseDirectory:** name of the directory that contains folders for every library to be loaded.
- **library:** name of the library to be loaded. There must be a folder with that name into the *baseDirectory*.
- **errorsDir:** folder where the status files for the library loading will be created.

---

## Necessary files

Into the folder for every library to be loaded must be these folders and files:

- **df.** It contains data to be loaded and format for files, in these folders.
  - **data.** For every member of every file to be loaded there must be a file named: *FileName\_MemberName.unl* (or *FileName\_FileName.unl* if file hasn't members). This file contains data to be loaded into the database table of the file.

Format for the UNL file is: every record in a line, every field in the record is delimited by character "|" (ASCII 124).

These files are obtained at conversion time, but user can create its own procedure to obtain these files with data to be loaded.

- **dds.** It contains, for every file to be loaded and for every logical file, a XML file with its format definition. This files are obtained at conversion time.
  - **lf.** It contains information of the members that composes the logical files.
  - **pf.** It contains information of the members that composes the physical files.
  - **sql.** It contains, for every physical file, the SQL information in order to create the database table.
- **dtaara.** Definition of the library data areas, a XML file for every one.

---

## How this task works

First of all locate the folder where the information for the library is, then for every physical file to be created:

- Creates a table into the database for every member of the file, using the information at *df/sql* folder.
- Loads data into the table, reading UNL file at *df/data*.
- Inserts information about file into Caravel catalog tables:
  - A record in *sysdf* to register the physical file.
  - A record in *syspfmbr* to register every member of the physical file.
  - In *sysmodel* loads the information of XML files for format file (at *df/dds*) and sql information (at *df/sql*). This information will be used if in the application configuration the property *manageModels* is true.

For every logical file:

- If the file has a key defined, it's created a new index for the table. If the file select data from other files, isn't converted into a database item (a view or a table). Caravel manages the catalog information to select data from the correct files when they are used.
- Inserts information about file into Caravel catalog tables:
  - A record in *sysdf* to register the logical file.
  - A record in *syslfmbr* to register every member of the logical file and its key components (if the file has a key).
  - In *sysmodel* loads the information of XML file for format file (at *df/dds*). This information will be used if in the application configuration the property *manageModels* is true.

---

## Chapter 6. Execution

Work directory must be where XML files for Ant tasks are. There, you'll execute:

- To create and load entities tables:

```
Ant -buildfile entityDataLoader.xml
```

If an error is produced, a file will be created in `\data\errors`.

- To create application tables:

```
Ant -buildfile CreateCatalog.xml
```

- To load application tables:

```
Ant -buildfile LoadData.xml
```

If an error is produced, a folder *errors* will be created in every folder for the libraries loaded.

If the process for application database is wrong and doesn't end successfully, then it's necessary to execute Ant with `CleanDatabase.xml`, that delete all tables in application database.

```
Ant -buildfile CleanDatabase.xml
```

---

### After execution

Modify at `system.xml.properties` file the property `updateEntitiesVersion`, setting value "false".

Execute `ExecutionSwing.bat` to access to Caravel Command Line and manage entities system information.

User QSECOFR (with no password) hasn't any initial program and access to command line. Then you can use `WRKUSRPRF USRPRF(*ALL)` to set the password for the users.