



# Contents

<b>The Object-Relational Force4 E/R Modeler</b>	<b>2</b>
<b>Force4-Modeler Functions Summary</b>	<b>2</b>
Model	2
Entities and Attributes	2
Relations	2
Generating Primary Keys	3
<b>Functions</b>	<b>4</b>
General functions at model level	4
Reverse Engineering	5
Model-Import	6
Model-Export	7
Force4 Server Components	8
DDL-Script	9
General Functions at Entity, Attribute and Relational Level	11
<b>Description of the Editing Windows</b>	<b>12</b>
Model	12
Entities	14
Attributes	15
Relations	18
Generating Primary Keys	20

## The Object-Relational Force4 E/R Modeler

is an E/R modelling tool which serves to save data structures in a repository in order to be able to access up-to-date data during the modelling process. Furthermore, the Force4 Modeler acts as a developer interface between the database and JAVA i.e. Macromedia Flex with the relevant nomenclature.

The model represented in the Force4 modeler is initially an 'abstracted view', composed of entities, attributes and relations, in order to assemble and present the data structures resulting, for example, from a business process. The assembly can be performed manually in the modeler. The import of structures from existing databases (i.e. Reverse Engineering) via a direct JDBC connection is also possible.

If a model has been constructed according to the relevant functional guidelines, then databases and direct connections to JAVA and Macromedia Flex can be established.

Entities and attributes are the abstract counterparts to tables and columns in a database. In addition to the properties necessary to define a database table, further properties can be applied to entities and attributes in order to create specific compatibility to JAVA and Macromedia Flex.

Relations in the Force4 Modeler are the connections between the entities. The assembly of these relations produces a relational structure from a collection of entities. The view of the relations with respect to the database can be demonstrated by the construction of so-called foreign key constraints, conditions at database level, which present the relationships of the tables to each other.

## Force4-Modeler Functions Summary

### Model

In order to be identified, a model always requires a name and the so-called "Connection Dictionary" which is composed of the connection URL of the JDBC adapter with the relevant Log-in data and also the necessary drivers and plug-ins.

In turn, the connection dictionary is composed of the driver information which are applying to the relevant database and JDBC adapter, as well as those datatypes present in the database. The datatypes are then set at attribute level and play an important role when generating DDL scripts.

The main advantage of the Force4 modeler is the independent nature of the model from the current database adapter. An alteration to adapter data will initiate an analysis of the new adapter and an automatic adjustment of the datatypes. A manual adjustment to model data is rarely necessary.

By implementing functions such as duplication, the ability to copy entities, attributes and relations from one model to another, simple alterations as well as the efficient import and export of adapters, and the intuitive user interface, the administration of data models within Force4 requires a minimum of time and effort.

### Entities and Attributes

As a component with content, the table-defining entity with its attributes forms the basis for a data structure. Aside from the name and fringe parameters for the external representation of the structure of the database (external name, external datatype), internal properties are maintained which define the connectivity to JAVA and Macromedia Flex (internal datatype, internal name, attribute visibility, authorisations) and properties applying to both representations (primary keys, NULL values, column definition).

### Relations

The database structure is completed via the connecting elements, the relations, which are built upon the entities of a model. Within a relational database model, tables are associated with so-called master-detail relations. For a master data set many detail sets are linked together in 1/n relations (1 master / n (numerous) details) via a primary key column in the master table to a foreign key column in the detail table. In the Force4 Modeler, these relations between entities can be

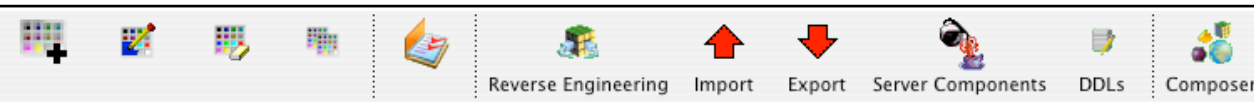





created. The database-specific implementation and the generation of the necessary JAVA and Macromedia Flex elements is carried out by the Force4 Modeler.

### **Generating Primary Keys**

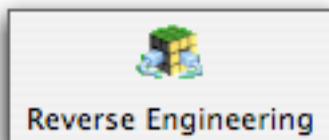
Aside from the possibility of generating the database primary key support via a DDL script and using such as standard, the Force4 Modeler is able to administer database-independent primary key support and generate the necessary JAVA elements. On the model, entity and attribute level, all methods together with the parameters of such methods can be set globally but also specifically at each level.

## Functions

### General functions at model level

	
	<p><b>New</b> Creation of a new empty model, either by providing a new adapter or by selecting an existing JDBC driver.</p>
	<p><b>Edit</b> Editing the current selected model.</p>
	<p><b>Delete</b> Deleting the current selected model.</p>
	<p><b>Duplicate</b> Duplicating the current selected model.</p>
	<p><b>Presettings</b> Modeler-specific presets for Reverse Engineering and the creation of DDL-scripts.</p>

## Reverse Engineering



Creating a model from an existing database via a JDBC connection

Step 1: In order to implement a Reverse Engineering model from an existing database, the following information must be known: the connecting URL, in order to address the database via JDBC, the username and password for the database connection and, if necessary, additional JDBC drivers or plugins for the JDBC connect in use.

Step 2: Selection of the model schemes and database object types to be engineered, with the possibility of limiting the number of database objects by means of a search pattern and the possibility of loading stored procedures. This information comes directly from the database, queried via the JDBC adapter.

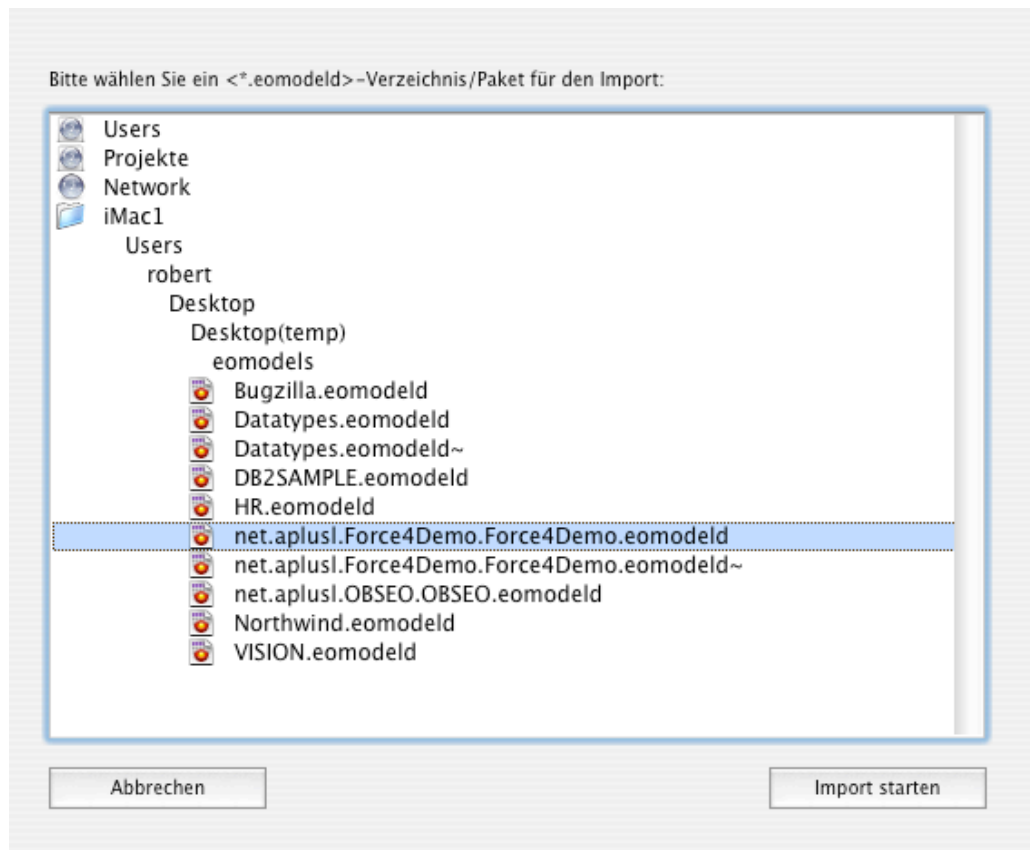
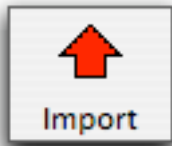
Step 3: Selection of the database objects and procedures to be engineered: A model may consist of all elements of the database or only a part thereof.

Step 4: Selection of the Reverse Engineering settings.

Reverse Engineering Settings	
<input checked="" type="checkbox"/>	<b>Add inverse relations to model</b> The "ToMany" relations present in the database are adopted within the model.
<input checked="" type="checkbox"/>	<b>Add relations to model</b> The "ToOne" relations present in the database are adopted within the model.
<input checked="" type="checkbox"/>	<b>Add schema names to table</b> The table names are imported into the entity column "external name" in the form <SCHEMA.TABLE>.
<input checked="" type="checkbox"/>	<b>Add all Attributes as &lt;visible&gt;</b> All entity attributes are imported into the model as visible, irrespective of whether they are visible in the database or not.
<input checked="" type="checkbox"/>	<b>Use custom entity classnames</b> Model-related classnames are created for the entities.
<input checked="" type="checkbox"/>	<b>Use minimum attributes for locking</b> Those entities with a minimum number of locking attributes (attributes which are part of an Update-WhereClause) are imported. Normally only the primary key is marked as a locking attribute.

## Model-Import

Importing a model which is to be found within the file system (normally in form of a ".eomodeld" package or folder)

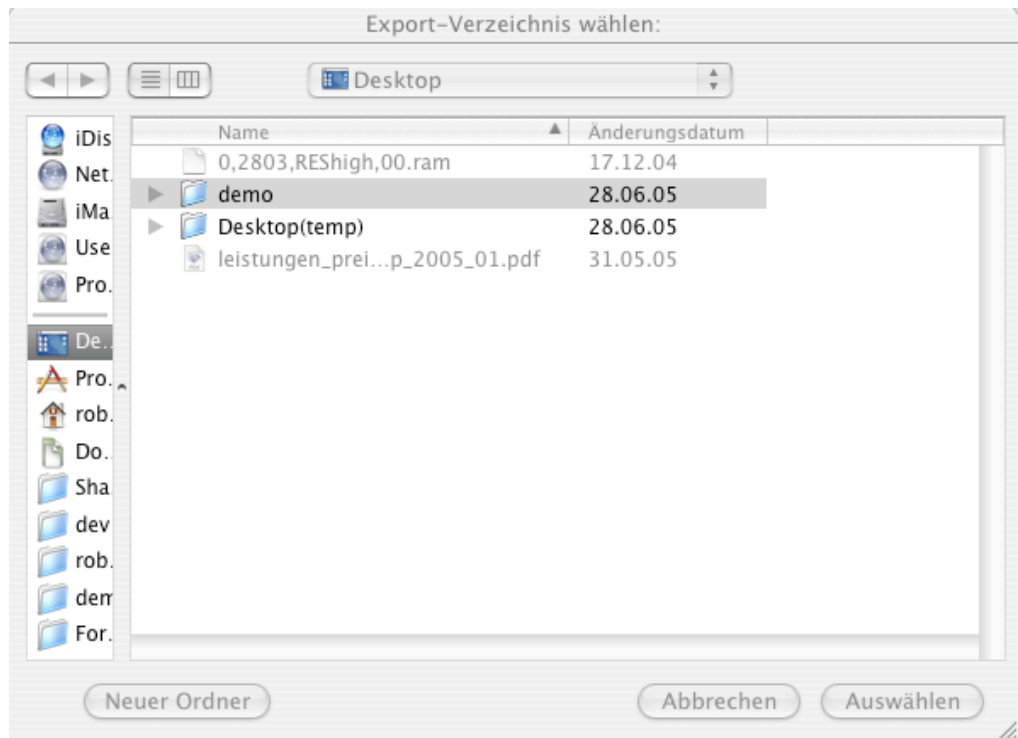
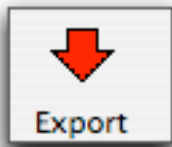


If the model name, present within the index file of the model being imported, has already been allocated, the user will be informed accordingly together with three choices: abort, new identifier (model name, package name) or replace current model.

If the user decides to allocate a new name then, he will be asked firstly for a new model name and secondly for a new package name.

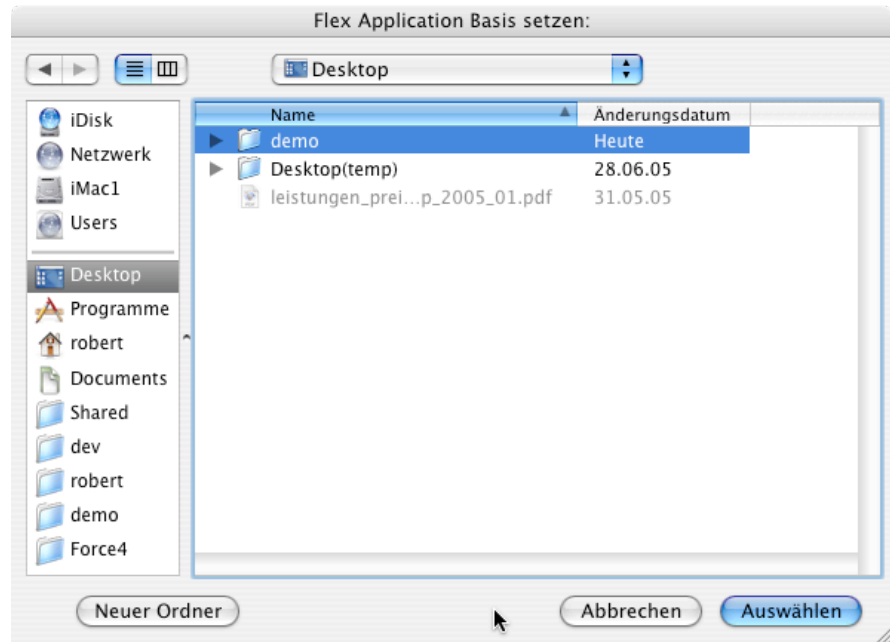
## Model-Export

Exporting the current selected model to the file system.



## Force4 Server Components

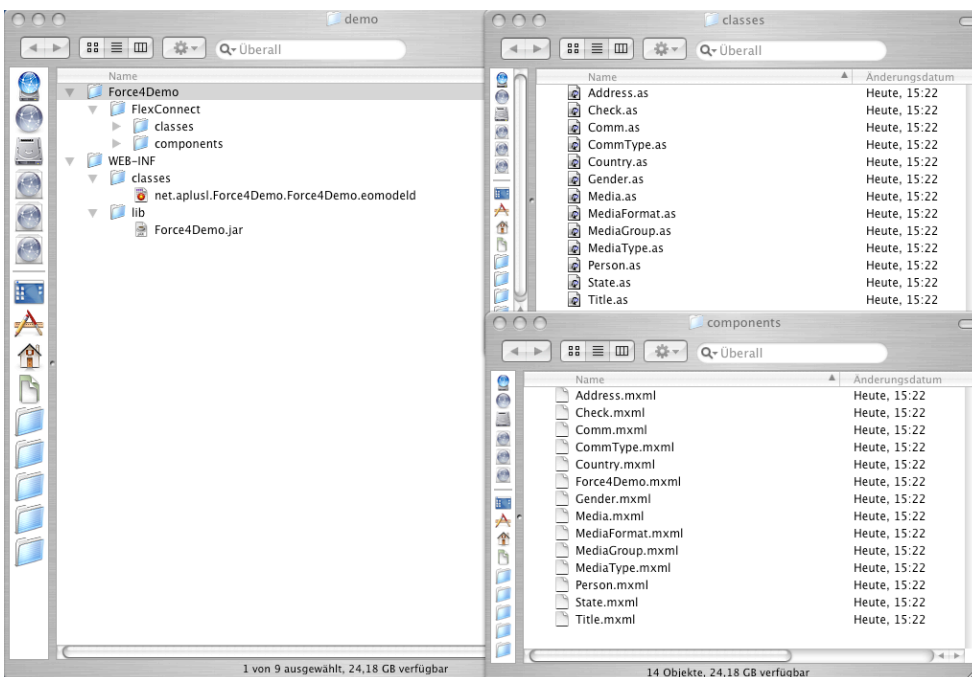
Generating the necessary components for Macromedia Flex and the Force4 server from the the current selected model.



Server Components – Step 1: File dialog to select the Force4/Flex server. The root folders for the Flexserver and the Force4 server are preset within the file dialog. The preset is derived from the Composer presetting or, if an alternative Force4 server is used, from the model.

Server Components – Step 2: Request to stop the the Force4 server. In order to generate the JAVA part of the Force4 server, the server with the file to be generated must be stopped.

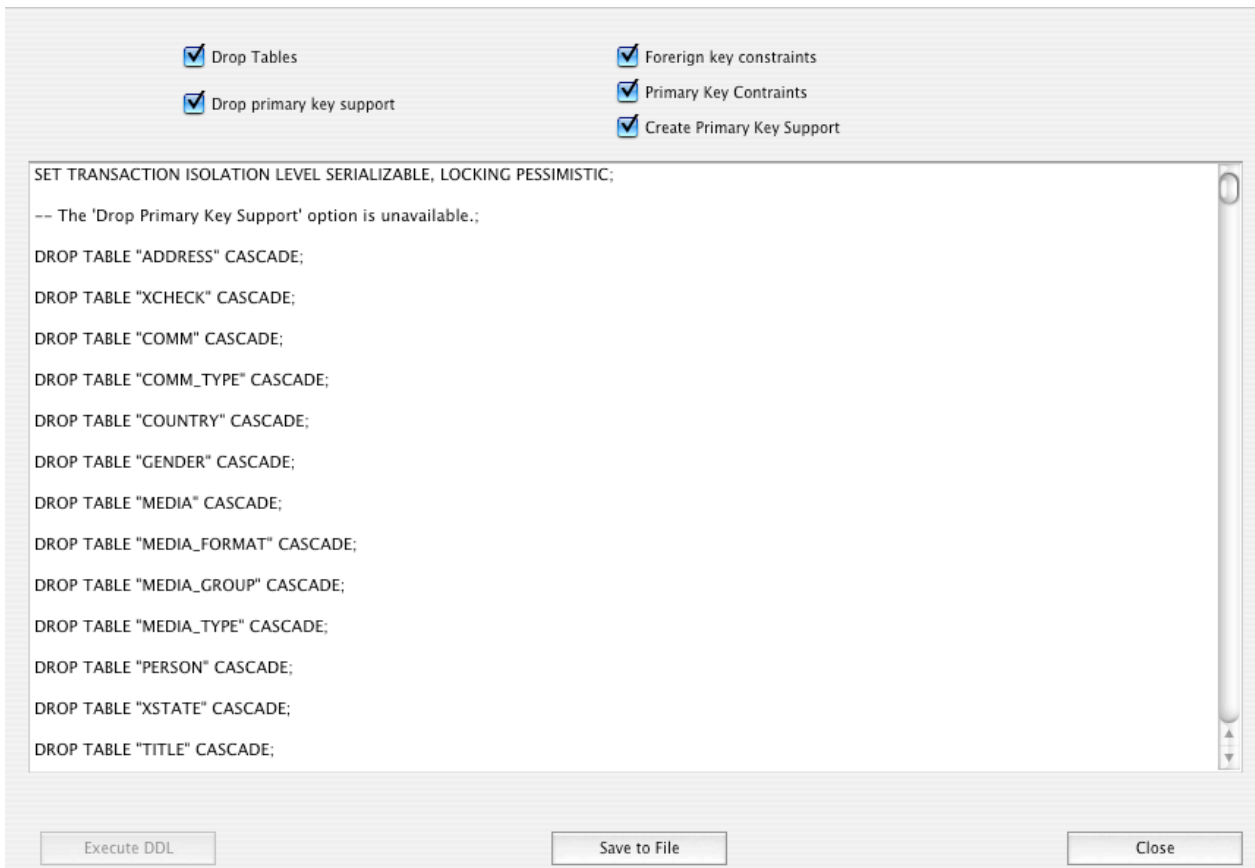
After successful generation the user will be informed and be prompted to restart the Force4 server.



Server Components – Result: A glimpse of the file system and the generated files.

## DDL-Script

Generating DDL scripts from the current selected model with the possibility of storage in the file system and the direct execution.



DDL Script: The script for generating the related database for the current selected model is generated here.

If "Execute DDL" is selected, the current visible script is sent to the database via JDBC adapter.

DDL Optionen	
<input checked="" type="checkbox"/>	<b>Drop Tables</b> Tables are deleted before generation.
<input checked="" type="checkbox"/>	<b>Drop Primary Key Support</b> For databases which support this function, primary key support is deleted before tables are created (e.g. deletion of sequences for Oracle databases).
<input checked="" type="checkbox"/>	<b>Primary Key Constraints</b> Primary key constraints are added to the table columns, which have a primary key attribute in the model.

DDL Optionen	
<input checked="" type="checkbox"/>	<b>Foreign Key Constraints</b> Foreign key constraints resulting from the relations are generated.
<input checked="" type="checkbox"/>	<b>Create Primary Key Support</b> For databases which support this function, database-side support for the generation of primary keys is generated (e.g. Sequences for Oracle databases).

## General Functions at Entity, Attribute and Relational Level

This context menu can be found in the entity/stored procedure list, the attribute/parameter list and the relations list. The menu items correlate, more-or-less, with the toolbar on the right side of the Editing window:

<b>Add</b>	⌘N
Copy	⌘C
Paste	⌘V
Duplicate	⌘D
Remove	⌘X
Save	⌘S
Undo	⌘Z

<b>Add</b> A new object is created (according to window entity/attribute ,,)
<b>Copy</b> Current object is copied to Force4 cache memory (this item exists only within this menu).
<b>Paste</b> Current content of Force4 cache memory is inserted into the list (this item exists only within this menu)
<b>Duplicate</b> Duplicate current object.
<b>Remove</b> Remove current object.
<b>Save</b> Save current status.
<b>Undo</b> Return to last saved status.

## Description of the Editing Windows

### Model

<b>Custom Name</b> <input type="text" value="Force4Demo"/>	
<b>Model Name</b> <input type="text" value="Force4Demo"/>	<b>Connection URL</b> <input type="text" value="jdbc:FrontBase://koloss.a-line.de/Force4Demo/isolation=repeatable_read/locking=optimistic"/>
<b>Package Name</b> <input type="text" value="net.aplusl.Force4Demo"/>	
<b>JDBC Driver</b> <input type="text"/>	<b>Initial Database SQL Statement</b> <input type="text"/>
<b>JDBC Plugin</b> <input type="text"/>	
<b>Username</b> <input type="text" value="force4"/>	<b>Primary Key Generation</b> Global Procedure <span style="float: right;">Speci</span> <input type="text" value="Default"/>
<b>Password</b> <input type="text" value="****"/>	
<input checked="" type="radio"/> Use Flex Server as Force4 Server <input type="radio"/> Use alternate Force4 Sever	Parameter <input type="text"/> Value <input type="text"/>
	<b>Current JDBC Driver</b> <input type="text" value="FrontBase - FBDriver - 2.3"/>

#### Custom Name

The visible model label in the Force4 system. Shown in the model tree and in the Composer as a field selection.

#### Model Name

The designation of the model for the server representation. The official model name used for the Flex and Java generation together with the package name and is subject to certain restrictions regarding the choice of characters. Allowed are alpha-numeric characters, however, the first letter of a name must not begin with numeral.

#### Package Name

The package name, together with the model name, forms a clearly-defined label within the system. This is required to identity the model within the system. If the model name is already clearly-defined in the system, then the package name can be left blank. Usual package names are (analog to JAVA) a combination of the inverse domain plus the model name. (net.aplusl.Force4Demo). However, any other name is acceptable, as long as it is unique within the system.

<b>JDBC Driver</b> If an additional driver is required for the JDBC adapter, then the driver name can be inserted here. Under normal circumstances, the standard driver, indicated by the connection URL , will suffice.
<b>JDBC Plugin</b> If an additional plugin is required for the JDBC adapter, then the plugin name can be entered here. Under normal circumstances, the standard driver, indicated by the connection URL , will suffice.
<b>Username</b> The database username with which the JDBC adapter references the database.
<b>Password</b> The database password with which the JDBC adapter references the database.
<b>Connection URL</b> Apart from the database-specific JDBC string, the JDBC connection URL (jdbc:<databaseconnect>:) includes details regarding the database host and the respective database. E.g. for Oracle: "jdbc:oracle:thin:@ORACLE-HOST:ORACLE-PORT:ORACLE-SID" or for Frontbase: "jdbc:Frontbase://FB-HOST/FB-DBName/"
<b>Initial Database SQL Statement</b> Should the JDBC adapter require further SQL statements in order to initialise the session after login, then these can be entered here.
<b>Use Flex Server as Force4 Server</b> The Macromedia Flex Server has its own JAVA server, which is used as standard. The files used by Force4 are generated into the Flex server structure. The Flex-specific MXML and ActionScript files are always integrated independently within the Flex architecture.
<b>Use alternate Force4 Server</b> Should one have an alternative JAVA server to execute the Force4 components, then this can be entered here.
<b>Force4 Server root</b> Should one be using an alternative Force4 server, then this will be the Force4 root folder, into which the server components will be generated.
<b>Force4 Server Endpoint URL</b> An alternative Force4 Server requires a URL which points to the so-called endpoint of the server.

## Entities

13 Entities of Model Force4Demo				
Name		External Name	Restricting Qualifier	Batch Faulting Size
Address		ADDRESS		
Check		XCHECK		
Comm		COMM		
CommType		COMM_TYPE		
Country		COUNTRY		
Gender		GENDER		
Media		MEDIA		
MediaFormat		MEDIA_FORMAT		
MediaGroup		MEDIA_GROUP		
MediaType		MEDIA_TYPE		
Person		PERSON		
State		XSTATE		
Title		TITLE		

**Entity: Address**

Address   Read Only ADDRESS   0

**Primary Key Generation**

Global Procedure

Specific Procedure

Parameter

Value

**External Query**

### Name

Name of the Entity: must be specific; must not be the same as the model or a contained relation.

### Read Only

If activated, then the entity can be read only. Inserting, deleting and updating is not permitted.

### External Name

Name of the external representation of the entity; in general the table name of the respective database table.

### Restricting Qualifier

Fields in the entity table can take limiting conditions, which are regarded when the entity is selected. E.g. mandatory dependent selection (ORGANISATION\_ID=4711).

### Batch Faulting Size

Number of single errors reported should the query fail. The download of follow-up error reports impose a burden en-route to the adapter, which can be limited here. Evaluation of collected error reports is thus deferred to business logic.

### External Query

An external query to an entity can be defined, which is called, if an unqualified query exists for this entity.: Example: the entity is queried without restriction.

Instead of:

„SELECT \* FROM ENTITY“ it is possible to enter here:

„SELECT NAME, PHONE, STREET, TOWN FROM ENTITY WHERE ORG\_ID=4711“

## Attributes

8 Attributes of Entity Address

Name	External Name	Internal Type	External Type	Width	Precision	Scale
checkId	CHECK_ID	Integer (i)	INTEGER	31		
city	CITY	Character	VARCHAR	1000		
companyName	COMPANY_NAME	Character	VARCHAR	300		
countryId	COUNTRY_ID	Integer (i)	INTEGER	31		
id	ID	Integer (i)	INTEGER	31		

Attribute: checkId

checkId  Primary Key  Used for locking  Visible  Allows Null  Read Only

Read Format:

Write Format:

Integer (i) INTEGER 0 31 0

Primary Key Generation

Global Procedure: Default Entity Procedure: Default

Specific Procedure:

Parameter:

Value:

### Name

Name of the attribute: must be specific in the entity; must not be the same as a relation of the entity.

### Primary Key

If activated, then this attribute is a primary key.

### Visible

If activated, then this attribute is visible and can be processed.

### Used for Locking

If activated, then this attribute will be used as an identifier in the UPDATE-WhereClause, should an Update take place.

### Allows Null

If activated, then a "NULL" value is allowed for this attribute.

### Read Only

If activated, then this attribute can be "read only". Changes to values are not permitted.

### External Name

Name of the external representation of the attribute, in general the column name of the respective database table.

### Internal Type

For internal processing of the attribute, the internal data type is used in Enterprise Objects, Java and Macromedia Flex. This is made up of a base type (Java Class) and a sub type (ValueType). The following values are possible:

Internal Type	Java Class	Value Type	
Character	java.lang.String		The Standard Type Character without ValueType validates automatically the column content to length with regard to the maximum number of characters for a varchar field in the database and selects the JDBC method "setString" if the content is shorter and "setCharacterStream" if longer.
Character (S)	java.lang.String	S	Always uses the JDBC method "setString".

Character (C)	java.lang.String	C	Always uses the JDBC method "setCharacterStream".
Character (E)	java.lang.String	E	Converts the text into raw UTF-8 and sends the content via the method "setBinaryStream" to a binary field in the database. It is not advisable to select this type, as it is normally up to the database to use the correct format when storing data.
Character (c)	java.lang.String	c	Frees up the text from appended empty spacing. Important for the external datatype CHAR, which uses empty spacing to fill column spaces, and should be used if these spaces do not have a specific function.
Integer(i)	java.lang.Number	i	Represent the Java sub types: java.lang.Integer, java.lang.Byte, java.lang.Double, java.lang.Short, java.lang.Float, and java.lang.Boolean via the abstract class: java.lang.Number.
Byte	java.lang.Number	b	
Double	java.lang.Number	d	
Short	java.lang.Number	s	
Long	java.lang.Number	l	
Float	java.lang.Number	f	
Boolean	java.lang.Number	c	
Integer (B)	java.lang.Number	B	Uses java.lang.Integer as Integer(i), however with a ValueType of B and greater precision (e.g. 31 places).
BigDecimal	java.math.BigDecimal		Are the types for decimal numbers with the possibility of defining the number of decimal places via "Scale". Depending upon the JDBC adapter in use, they either have the ValueType B or no ValueType.
BigDecimal(B)	java.math.BigDecimal	B	
Timestamp	com.webobjects.foundation.NSTimestamp		The TimeStamp without ValueType uses the standard JDBC methods "setObject" and "getObject" in order to transfer data. It is assumed that the database has at its disposal a format which is compatible to java.sql.timestamp.
Timestamp(D)	com.webobjects.foundation.NSTimestamp	D	Uses the methods "setDate" and "getDate" in the "java.util.Date" object.
Timestamp(t)	com.webobjects.foundation.NSTimestamp	t	Uses the methods "setTime" and "getTime" in the "java.sql.Time" object.
Timestamp(T)	com.webobjects.foundation.NSTimestamp	T	Uses the methods "setTimestamp" and "getTimestamp" in the "java.sql.Timestamp" object.
Timestamp(M)	com.webobjects.foundation.NSTimestamp	M	Is used instead of Timestamp(D) within MS-SQLServer. Only the "java.sql.Date" object is supported.
Binary Data	com.webobjects.foundation.NSData		

### **External Type**

This selection field holds all external datatypes which can be transferred to the database via the JDBC adapter. The datatypes are dependent upon the database and thus vary from adapter to adapter. Should one change the adapter of one model, then the datatypes will be adapted to the new database.

### **Width**

The maximum number of ASCII characters for a character attribute can be entered here.

### **Precision**

Total number of numerical characters in a numerical attribute.

### **Scale**

Total number of decimal places in a numerical attribute.

### **Read Format**

The content of attributes can, per default at adapter level, be converted for display purposes. For this to function the wild-card "%P" is used. If, for example, the content of a column is a time period, given in seconds, then it is possible to display such values as minutes using the format "%P /60".

### **Write Format**

The content of attributes can, per default at adapter level, be converted for storage purposes. For this to function the wild-card "%V" is used. If, for example, the content of a column is a time period, given in seconds, then it is possible to enter such values as minutes using the format "%V \*60". The value is stored in seconds.

## Relations

2 Relations of Entity Comm					
Name		Delete Rule	Join Type	Source Attribute	Destination Attribute
commType		nullify	inner	commTypeld	id ( CommType )
person		nullify	inner	personld	id ( Person )

Relation: commType					
commType		<input checked="" type="checkbox"/> Visible	nullify	inner	commTypeld
Batch Faulting Size	<input type="text" value="0"/>	<input type="checkbox"/> Mandantory	cascade	full outer	entry
		<input type="checkbox"/> ToMany Relation	deny	left outer	id
		<input type="checkbox"/> Owns Destination	no action	right outer	personld
		<input type="checkbox"/> Propagate Primar			CommType
					id
					name

### Name

Name of the relation: must be specific in the entity; must not be the same as the entity or a attribute of the entity.

### Visible

Relations are, per default, always visible. Invisible relations are not instantiated within Java and cannot be implemented.

### Mandantory

If a relation is a mandatory field, then the source dataset and target dataset must always be filled with valid values. I.e. NULL values are not permitted.

### ToMany Relation

In a "ToOne" relation there exists exactly one dataset on the source side to one dataset on the target side. In a "ToMany" relation there may exist one, many or no datasets on the target side.

### Owns Destination

Only the parent entity has the proprietary right over the child entity in a parent-child relationship. If the relation is deleted, then the child entry is also automatically deleted.

### Propagate Primary Key

If the primary key in a parent-child relationship is propagated, then the primary key generation for the child entry is interrupted and the primary key of the parent is adopted.

### Delete Rule

Possible values:

- Nullify - Should the parent entry be deleted, then the child foreign key is set to NULL.
- Cascade - Deletes all child entries together with the parent entry.
- Deny - A parent entry cannot be deleted, if child entries are still connected via this relation.
- Do nothing - Parent-entries are deleted without affecting child entries. This could infringe upon the data integrity and should be implemented with care.

### Join Type

Possible values:

- Inner Join - Source datasets for relations are only displayed if the target datasets exists.
- Full Outer Join - All source datasets are displayed.
- Left Outer Join - Source datasets without target datasets are saved before deletion.
- Right Outer Join - Target datasets without source datasets are saved before deletion.

### **Source Attribute**

The relations source attribute.

### **Destination Attribute**

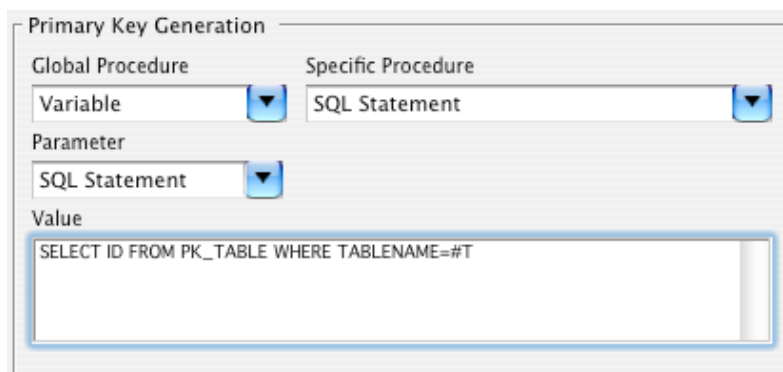
Entity and attribute which are pointed to by the relation.

### **Batch Faulting Size**

Number of single errors reported should the dissolving of a ToMany relation fail. The download of follow-up error reports impose a burden en-route to the adapter, which can be limited here.

Evaluation of collected error reports is thus deferred to business logic.

## Generating Primary Keys



Primary Key Generation

Global Procedure      Specific Procedure

Variable      SQL Statement

Parameter

SQL Statement

Value

```
SELECT ID FROM PK_TABLE WHERE TABLENAME=#T
```

### Global Procedure:

Setting the procedure for generating primary keys: Selection of the general procedure (valid for the whole model). Possible settings are:

- Default – The procedure of the JDBC adapter is used.
- SQL Statement – A SQL statement entered in “Value” relevant to the parameter “SQL statement” serves as primary key.
- JavaClass – A Java class exists which takes over the primary key generation.
- Oracle Sequence – For an Oracle database, Oracle sequences can be set for generation.
- Variable – Globally no procedure for primary key definition is defined. Specific procedures apply.

### Specific Procedure:

Global procedure is variable → specific procedures for generation can now be set on all levels:

During evaluation the system begins to search for the specific procedure at attribute level. If this has not been set, the system will look into the relevant entity and then in the relevant model. The parameters of the procedure are checked analogue, but independent from the procedure. If a parameter value has not been set, then the value from the next step in the hierarchy is adopted. If no value is found then the adapter will perform with standard methods. Findet sich abschließend kein Wert so greift das Standard-Verhalten des Adapters auch wenn spezifische Methoden eingetragen sind.

### Parameter

Possible parameters are "SQL statement", "Java Class" and for Oracle databases "Oracle Sequence". These parameters are set permanently for the respective global procedures. If "variable" should be set here, then values can be set for all parameters.

### Value:

Values for parameters are entered here.

### Parameter SQL Statement:

A SQL statement, which takes over the generation of primary keys. The following wild-cards are valid for the statement: #T for the table, #C for the column.

### Parameter JavaClass:

A Java class, which takes over the generation of primary keys. Within this class key methods can be found which the system will implement. A description of the APIs will follow.

### Parameter Oracle Sequence:

A sequence, which takes over the generation of primary keys. The following wild-cards can be implemented within the sequence name: #T for the table, #C for the table column.