

Transfer Configuration File

This configuration file defines the packages and objects that are created and produced by Transfer.

The xml schema file can be found at: </transfer/resources/xsd/transfer.xsd>.

It is **highly recommended** that when creating a new transfer xml file, you validate your xml against this xml schema.

Items in *italics* denoted an element attribute.

Elements

- **transfer**

Top level element for the xml file.

- **includes** (optional)

Configure any include files that are used in the configuration of Transfer

- **include**

This is an include of a Transfer configuration file.

path The relative path from your application root to the include file.

overwrite (optional) Whether or not to overwrite the previous configurations. Defaults to 'false'.

- **objectCache** (optional)

Configure the settings for all object caching parameters. If this is not set, the default values are used. More details on caching can be found at [Managing the Cache](#)

- **scopes** (optional) Defines the keys that Transfer stores it's caching when using caching in shared scopes. If this is not set, the default values are used.

- **application** (optional)

The scope key for 'application' cached objects, if not set, all caching is done under the 'application.transfer' scope.

key The key to set the cache under, e.g. 'transfer'

- **session** (optional)

The scope key for 'session', and cached objects, if not set, all caching is done under the 'session.transfer' scope. This also applies to 'transaction' scoped objects, as these caches share these scopes.

key The key to set the cache under, e.g. 'transfer'

- **request** (optional)

The scope key for 'request' cached objects, if not set, all caching is done under the 'request.transfer' scope.

key The key to set the cache under, e.g. 'transfer'

- **server** (optional)

The scope key for 'server' cached objects, if not set, all caching is done under the 'server.transfer' scope.

key The key to set the cache under, e.g. 'transfer'

- **defaultcache** (optional)

This sets the default values for caching of objects within Transfer. If this is not set, the default values are used.

 - **maxobjects** (optional)

The maximum number of objects to store for any one Object class.
value The number of TransferObjects to cache, defaults to unlimited (0).
 - **maxminutespersisted** (optional)

The maximum number of minutes to store any TransferObject of a given class.
value The number of minutes to cache, defaults to unlimited (0).
 - **accessedminutestimeout** (optional)

The number of minutes in which a TransferObject will timeout in, if it has not been accessed.
value Number of minutes until cache timeout, defaults to having no effect (0).
 - **scope** (optional)

The scope to store all TransferObjects under. Details on the types of caching available can found at [Managing the Cache](#)
type 'instance', 'application', 'session', 'request', 'transaction', 'server', 'none'. Defaults to 'instance'
- **cache** (optional)

class The class to set the caching settings for
This overwrites the default settings for caching for a specific class

 - **maxobjects** (optional)

The maximum number of objects to store for this class.
value The number to cache, defaults to unlimited (0).
 - **maxminutespersisted** (optional)

The maximum number of minutes to store TransferObjects of this class.
value The number of minutes to cache, defaults to unlimited (0).
 - **accessedminutestimeout** (optional)

The number of minutes in which a TransferObject will timeout in, if it has not been accessed.
value Number of minutes until cache timeout, defaults to having no effect (0).
 - **scope** (optional)

The scope to store TransferObjects of this class under. Details on the types of caching available can found at [Managing the Cache](#).
type 'instance', 'application', 'session', 'request', 'transaction', 'server', 'none'. Defaults to 'instance'
- **nullValues** (optional)

Configure the default null values for properties

 - **string**

The default null value for all string types.
value The null value to use. Defaults to "".
 - **numeric**

The default null value for all numeric types.

value The null value to use. Defaults to 0.

- **date**

The default null value for all date types.

value The null value to use. Defaults to 1/1/100.

- **boolean**

The default null value for all boolean types.

value The null value to use. Defaults to 'false'.

- **UUID**

The default null value for all UUID types.

value The null value to use. Defaults to '00000000-0000-0000-0000000000000000'.

- **GUID**

The default null value for all GUID types.

value The null value to use. Defaults to '00000000-0000-0000-0000-000000000000'.

- **binary**

The default null value for all binary types.

value The null value to use (Whatever is written here is converted to a Java byte array). Defaults to empty byte array.

- **objectDefinitions**

All the Object definitions

- **package**

Organisational element to store various objects of similar aspect together. e.g. a 'user' package.

name The name of the package

- **object**

Package can contain multiple object elements

- **package**

Package can contain multiple package elements

- **object**

A class definition for a [transfer.com.TransferObject](#) to be manufactured into a bean that represents a record in the set database table.

name The name of the TransferObject, e.g. 'User'.

table (optional) The database table that the TransferObject refers to, e.g. 'tbl_User'. Defaults to the value of @name

decorator (optional) The value of a ColdFusion component that extends [transfer.com.TransferDecorator](#) and will decorate the generated TransferObject. Details on Decorators can be found at [Writing Decorators](#)

sequence (optional) Specifically for Oracle and PostGreSQL support. This is the name of the sequence that corresponds with this object's table. Defaults to the value of @table + '_seq'

tablealias (optional) This is an table alias that will be used for get() operations in Transfer. This is usually most useful for Oracle when the generated alias is over 30 characters.

- **id**

This is the definition of the primary key as set in the database.

This results in getter and setter functions on the

TransferObject. Details on all these methods can be found at [Ger](#)

Generated Methods.

There can only be one ID declaration, but either 'id' or 'compositeid' must be present.

name The name of the ID, e.g. 'IDUser'

type The typing of the primary key, either numeric, string, date, boolean, UUID, GUID, binary.

column (optional) The primary key column on the table of this TransferObject. e.g. 'IDUser'. Defaults to the value of @name.

generate (optional) If this is set to true, Transfer will generate the primary key when the record is inserted into the database. Otherwise it will assume that the primary key is automatically generated by the database. Transfer currently supports generating numeric, UUID, and GUID IDs. Defaults to false.

- **compositeid**

A composite key declaration, which contains any combination of foreign key relationships and/or property values. This results in a 'getCompositeID()' function on the TransferObject. Details on all these methods can be found at [Generated Methods](#). There can only be one compositeid declaration, but either 'id' or 'compositeid' must be present.

- **property (optional)**

The property to use in this composite id. The 'property' element can be used more than once.

name The name attribute matches the name attribute of a configured property within the object.

- **manytoone (optional)**

The manytoone relationship to use in this composite id. This represents the usage of a foreign key in the composite id. The 'manytoone' element can be used more than once.

name The name attribute matches the name element of a configured manytoone element within the object.

- **parentonetomany (optional)**

The parent onetomany relationship to use in this composite id. This represents the usage of a foreign key in the composite id. The 'parentonetomany' element can be used more than once.

class The name attribute matches the name element of a configured manytone element within the object.

- **property**

A single property on the TransferObject.

This results in getter and setter functions on the TransferObject. Details on these methods can be found at [Generated Methods](#).

Default values for these properties can be found in [Default Property Values](#)

TransferObjects can have multiple property declarations.

name The name of the property, e.g. 'Name'

type The type of the property. Either numeric, string, date, boolean, UUID, GUID, binary.

column (optional) The column on the table this property refers

to, e.g. 'user_Name'. Defaults to the value of @name
set (optional) If this is set to 'false' the setter function has a scope of private. Defaults to 'true'.

nullable (optional) Defines whether or not values from this property can be inserted into the database as NULL values.

nullvalue (optional) Overrides the default null value for this property. Only relevant if @nullable='true'.

ignore-insert (optional) Will ignore this property when performing an insert of this object into its table.

refresh-insert (optional) This will query the database for the value of this column and refresh its state after an insert.

ignore-update (optional) Will ignore this property when performing an update of this object into its table.

refresh-update (optional) This will query the database for the value of this column and refresh its state after an update.

- **manytoone**

This defines a relationship when many of a this object's table records link directly to one of another tables. e.g.

'tbl_User.lnkIDPermission = tbl_Permission.IDPermission'.

It creates getter and setter functions on the TransferObject that get and set the composite TransferObject. Details on these methods can be found at [Generated Methods](#).

TransferObjects can have multiple manytoone declarations.

You can only have one manytoone or oneotomany definition for each foreign key. You may also not share a foreign key used in a manytoone with a <property> definition

name The name of this relationship, e.g. 'Permission'.

lazy (optional) If this is set to 'true', the manytoone composition will only be loaded when requested. This defaults to 'false'. For more information, see [Using Lazy Loading](#).

proxied (optional) If this is set to 'true', only a proxy of the TransferObject will be loaded, rather than the full object data, for each TransferObject. Default is 'false'. For more information, see [Using TransferObject Proxies](#).

- **link**

Defines what other TransferObject the composition is made up of.

to The class name of the TransferObject this links to, e.g. 'user.Permission'.

column The column on the parent TransferObject table that is the foreign key to the composite child, e.g. 'lnkIDPermission'.

- **onetomany**

This defines a relationship where one of this object's table records corresponds to many of another tables records. e.g. 'tbl_User.IDUser = tbl_Post.lnkIDUser'.

This creates several methods on the parent and child TransferObjects, dependent on the set collection type. Details on all these methods can be found at [Generated Methods](#).

TransferObjects can have multiple one to many declarations.

You can only have one oneotomany or manytoone definition for each foreign key. You may also not share a foreign key used in a manytoone with a <property> definition

name The name of this collection, e.g. 'Permission'.

lazy (optional) If this is set to 'true', the onetomany composition will only be loaded when requested. This defaults to 'false'. For more information, see [Using Lazy Loading](#).

proxied (optional) If this is set to 'true', only a collection of proxies of the TransferObject will be loaded, rather than the full object data, for each TransferObject. Default is 'false'. For more information, see [Using TransferObject Proxies](#).

- **link**

Defines the TransferObject the composition is made up of.

to The class name of the TransferObject this links to, e.g. 'user.Permission'.

column The column on the child TransferObject table that is the foreign key to the parent, e.g. 'InkIDUser'.

- **collection**

This element sets the details of the type of collection of TransferObject that is added to the parent.

type Either 'struct' or 'array'. If 'struct' is selected, a structure is used to manage the composite

TransferObjects, whereas an array is used when the type is set to 'array'.

- **condition** (Optional) Used to place a filtering condition on the child elements that are retrieved from the database.

property (optional) The property on the child object to filter by.

value (required if property has a value) The value of the property to filter by.

where (required if property/value is not set) The SQL statement to filter the child object by. Child properties enclosed with '{ }' will be resolved to their column names.

- **key** (if *collection[@type]* = 'struct')

The property to be used in the key when adding and retrieving items from parent. This property must have a unique value.

property The name of the property on the child TransferObject to be used as the key, e.g. 'permissionName'.

- **order** (optional if *collection[@type]* = 'array')

The order in which to sort the collection.

property The name of the property on the child TransferObject to sort the array by.

order The order to sort by, either 'asc' or 'desc'.

- **manytomany**

This defines a relationship where many of this object's table

records corresponds to many of another tables. This is normally accomplished at a database level by use of an intermediary linking table. e.g. 'tbl_User.IDUser = lnk_UserPermssion.lnkIDUser AND lnk_UserPermission.lnkIDPermission = tbl_Permission.IDPermission'.

This creates several methods on the parent TransferObject, dependent on the set collection type. Details on all these methods can be found at [Generated Methods](#).

TransferObjects can have multiple manytomany declarations.

You can only have one manytomany definition for each linking table. Bi-directional manytomany is not yet supported.

name The name of this collection, e.g. 'Permission'.

table The name of the intermediary table that creates the many to many relationship. e.g. 'lnk_UserPermission'

lazy (optional) If this is set to 'true', the manytomany composition will only be loaded when requested. This defaults to 'false'. For more information, see [Using Lazy Loading](#).

proxied (optional) If this is set to 'true', only a collection of proxies of the TransferObject will be loaded, rather than the full object data, for each TransferObject. Default is 'false'. For more information, see [Using TransferObject Proxies](#).

tablealias (optional) This is an table alias that will be used for get() operations in Transfer. This is usually most useful for Oracle when the generated alias is over 30 characters.

- **link**

The first link specifies the connection information for the intermediary table to the parent table.

to This first link MUST be the same class name as the Parent TransferObject. e.g. 'user.User'

column This is the column on the intermediary table that is the foreign key to the parent TransferObject's table, e.g. 'lnkIDUser'

- **link**

The second link defines the TransferObject the composition is made up of.

to This second link is the class name to the child TransferObject, e.g. 'user.Permission'.

column This is the column on the intermediary table that is the foreign key to the child TransferObject's table, e.g. 'lnkIDPermission'

- **collection**

This element sets the details of the type of collection of TransferObject that is added to the parent.

type Either 'struct' or 'array'. If 'struct' is selected, a structure is used to manage the composite

TransferObjects, whereas an array is used when the type is set to 'array'.

- **condition (Optional)** Used to place a filtering condition on the child elements that are retrieved

from the database.

property (optional) The property on the child object to filter by.

value (required if property has a value) The value of the property to filter by.

where (required if property/value is not set) The SQL statement to filter the child object by. Child properties enclosed with '{ }' will be resolved to their column names.

- **key** (if *collection[@type] = 'struct'*)

The property to be used in the key when adding and retrieving items from parent. This property must have a unique value

property The name of the property on the child TransferObject to be used as the key, e.g. 'permissionName'.

- **order** (optional if *collection[@type] = 'array'*)

The order in which to sort the collection.

property The name of the property on the child TransferObject to sort the array by.

order The order to sort by, either 'asc' or 'desc'.

- **function**

A cfml function that you want added to this TransferObject.

TransferObjects can have multiple function declarations.

Details on custom methods can be found at [Custom Methods](#).

name The name of the function

access Access of the function, public, private or package.

returntype Type of what is returned by this function.

- **argument**

A defined argument for the function

A Function can have multiple function declarations.

name The name of the argument

type The type of the argument


required (optional) If the argument is required, defaults to 'false'.

default (optional) The default value for this argument if it is not set.

- **body**

The cfml body of the function. It may be necessary to utilise a CDATA declaration in this area.

There can only be one body declaration per function.

 Categories:

- [Configuration](#)