

Oracle® Data Integrator

Substitution Methods Reference

10g Release 3 (10.1.3)

September 2008

Copyright © 2006, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Table Of Contents

Organization of This Manual	1
Using Substitution Methods.....	3
The Principles	3
Generic Syntax	3
Specific Syntax for CKM	3
Using Flexfields	3
Using Substitution Methods in Actions.....	5
Introduction	5
Action Lines Code.....	5
Action Calls Methods	5
List of Substitution Methods	7
Global Methods.....	7
Journalizing Methods.....	7
Loading Methods	7
Check Methods.....	8
Integration Methods	8
Reverse Engineering Methods	9
Web Services Methods.....	9
Action Methods	9
Substitution Methods Reference	11
getAK() method.....	11
getAKColList() Method.....	11
getCatalogName() Method	14
getCatalogNameDefaultPSchema() Method	15
getCK() Method	17
getColDefaultValue() Method	18
getColList() Method	18
getColumn() Method.....	24
getContext() Method	26
getDataType() Method.....	27
getFilter() Method	29
getFilterList() Method.....	29
getFK() Method.....	31
getFKColList() Method.....	32
getFlexFieldValue Method	35
getFrom() Method	36
getGrpBy() Method	36
getGrpByList() Method.....	37
getHaving() Method	38

getHavingList() Method.....	39
getIndex() Method.....	40
getIndexColList() Method.....	41
getInfo() Method	43
getJDBCConnection() method.....	47
getJoin() Method	48
getJoinList() Method	48
getJrnFilter() Method	50
getJrnInfo() Method	50
getModel() Method.....	52
getNewColComment() Method	53
getNewTableComment() Method	54
getNotNullCol() Method	54
getObjectName() Method	55
getObjectNameDefaultPSchema() Method	57
getPK() Method.....	58
getPKColList() Method.....	58
getPop() Method	61
getPrevStepLog() Method.....	62
getQuotedString()	64
getOption() Method getUserExit() Method.....	65
getSchemaName() Method.....	65
GetSchemaNameDefaultPSchema() Method.....	66
getSession() Method.....	68
getSessionVarList() Method	69
getSrcColList() Method	69
getSrcTablesList() Method.....	72
getStep() Method	74
getSubscriberList() Method.....	76
getSysDate() Method.....	77
getTable() Method	77
getTargetColList() Method	80
getTargetTable() Method	84
getUser() Method.....	86
getOption() Method getUserExit() Method.....	86
hasPK() Method.....	87
isColAttrChanged() Method	87
nextAK() Method	88
nextCond() Method	89
nextFK() Method	89

setNbInsert, setNbUpdate, setNbDelete, setNbErrors and setNbRows Methods.....	90
---	----

This manual provides a reference of the **Oracle Data Integrator Substitution Methods**. It is intended for advanced developers who want to create generic knowledge modules or procedures using these methods in their integration scenarios.

Organization of This Manual

This manual contains the following:

- **Chapter 1 - Using the Substitution Methods** explains where and how to use substitution methods.
- **Chapter 2 - Substitution Methods Reference** provides a detailed syntax for each method.

Using Substitution Methods

The Principles

The methods that are accessible from the Knowledge Modules and from the procedures are direct calls to Oracle Data Integrator methods implemented in Java™. These methods are usually used to generate some text that corresponds to the metadata stored into the Oracle Data Integrator repository.

Generic Syntax

The substitution methods are used in any text of a task of a Knowledge Module or of a procedure. The syntax to use is the following:

Where:

- `Free text`: any text of a task in the language of the desired technology.
- `Java expression`: any Java expression that allows the construction of a string.

Java expression example:

```
odiRef.getTable("WORK_TABLE") + "FUTURE"
```

The Oracle Data Integrator API is implemented in the Java class `OdiReference`, whose instance `OdiRef` is available at any time. Thus, to call the Data Integrator method `getFrom()`, you have to write `odiRef.getFrom()`.

Note: The previous syntax `snpRef.<method_name>` is still supported but deprecated.

Specific Syntax for CKM

The following syntax is used in a IKM to call the execution of a check procedure (CKM).

This syntax automatically includes all the CKM procedure commands at this point of in the processing.

The options for this syntax are:

- `CKM_FLOW`: triggers a flow control, according to the CKM choices made in the Control tab of the Interface.
- `CKM_STATIC`: Triggers a static control of the target datastore. Constraints defined for the datastore and selected as Static constraints will be checked.
- `DELETE_ERRORS`: This option causes automatic suppression of the errors detected.

Using Flexfields

Flexfields are user-defined fields enabling to customize the properties of Oracle Data Integrator' objects. Flexfields are defined on the **Flexfield** tab of the object window and can be set for each object instance through the **Flexfield** tab of the object window.

When accessing an object properties through Oracle Data Integrator' substitution methods, if you specify the Flexfield **Code**, Oracle Data Integrator will substitute the **Code** by the flexfield value for the object instance.

For instance:

`<%=odiRef.getTable("L", "MY_DATASTORE_FIELD", "W") %>` will return the value of the flexfield MY_DATASTORE_FIELD for the current table.

`<%=odiRef.getSrcTableList("", "[MY_DATASTORE_FIELD] ", " ", " ") %>` will return the flexfield value for each of the source tables of the interface.

It is also possible to get the value of a flexfield through the `getFlexFieldValue()` method.

Important: Flexfield exist only for certain object type. Objects that do not have a flexfield tab do not support them.

Using Substitution Methods in Actions

Introduction

An action corresponds to a DDL operation (create table, drop reference, etc). Each action contains several **Action Lines**, corresponding to the commands required to perform the DDL operation (for example, dropping a table requires dropping all its constraints first).

Action Lines Code

Action lines contain statements valid for the technology of the action group. Unlike procedures or knowledge module commands, these statements use a single connection (SELECT ... INSERT statements are not possible). In the style of the knowledge modules, action make use of the substitution methods to make their DDL code generic.

For example, an action line may contain the following code to drop a check constraint on a table:

```
ALTER TABLE <%=odiRef.getTable("L", "TARG_NAME", "A") %> DROP CONSTRAINT
<%=odiRef.getCK("COND_NAME") %>
```

Action Calls Methods

The Action Calls methods are usable in the action lines only. Unlike other substitution methods, they are not used to generate text, but to generate actions appropriate for the context.

For example, to perform the a Drop Table DDL operation, we must first drop all foreign keys referring to the table.

In the *Drop Table* action, the first action line will use the `dropReferringFKs()` action call method to automatically generate a *Drop Foreign Key* action for each foreign key of the current table. This call is performed by creating an action line with the following code: `<% odiRef.dropReferringFKs(); %>`

The syntax for calling the action call methods is:

```
<% odiRef.method_name(); %>
```

Note: The action call methods must be alone in an action line, should be called without a preceding "=" sign, and require a trailing semi-colon.

The following Action Call Methods are available for Actions:

- **addAKs()**: Call the *Add Alternate Key* action for all alternate keys of the current table.
- **dropAKs()**: Call the *Drop Alternate Key* action for all alternate keys of the current table.
- **addPK()**: Call the *Add Primary Key* for the primary key of the current table.
- **dropPK()**: Call the *Drop Primary Key* for the primary key of the current table.
- **createTable()**: Call the *Create Table* action for the current table.
- **dropTable()**: Call the *Drop Table* action for the current table.
- **addFKs()**: Call the *Add Foreign Key* action for all the foreign keys of the current table.

- **dropFKs()**: Call the *Drop Foreign Key* action for all the foreign keys of the current table.
- **enableFKs()**: Call the *Enable Foreign Key* action for all the foreign keys of the current table.
- **disableFKs()**: Call the *Disable Foreign Key* action for all the foreign keys of the current table.
- **addReferringFKs()**: Call the *Add Foreign Key* action for all the foreign keys pointing to the current table.
- **dropReferringFKs()**: Call the *Drop Foreign Key* action for all the foreign keys pointing to the current table.
- **enableReferringFKs()**: Call the *Enable Foreign Key* action for all the foreign keys pointing to the current table.
- **disableReferringFKs()**: Call the *Disable Foreign Key* action for all the foreign keys pointing to the current table.
- **addChecks()**: Call the *Add Check Constraint* action for all check constraints of the current table.
- **dropChecks()**: Call the *Drop Check Constraint* action for all check constraints of the current table.
- **addIndexes()**: Call the *Add Index* action for all the indexes of the current table.
- **dropIndexes()**: Call the *Drop Index* action for all the indexes of the current table.
- **modifyTableComment()**: Call the *Modify Table Comment* for the current table.
- **AddColumnsComment()**: Call the *Modify Column Comment* for all the columns of the current table.

List of Substitution Methods

Global Methods

- `getCatalogName`
- `getCatalogNameDefaultPSchema`
- `getColDefaultValue`
- `getContext`
- `getDataType`
- `getFlexFieldValue`
- `getInfo`
- `getJDBCConnection`
- `getObjectName`
- `getObjectNameDefaultPSchema`
- `getOption`
- `getPrevStepLog`
- `getSchemaName`
- `GetSchemaNameDefaultPSchema`
- `getSession`
- `getSessionVarList`
- `getStep`
- `getSysDate`
- `getUserExit`
- `setNbDelete`
- `setNbErrors`
- `setNbInsert`
- `setNbRows`
- `setNbUpdate`

Journalizing Methods

- `getJrnFilter`
- `getJrnInfo`
- `getSubscriberList`
- `getTable`
- `getColList`

Loading Methods

- `getColList`
- `getFilter`
- `getFilterList`
- `getFrom`
- `getGrpBy`
- `getGrpByList`

- `getHaving`
- `getHavingList`
- `getJoin`
- `getJoinList`
- `getJRNFilter`
- `getJrnInfo`
- `getPop`
- `getSrcColList`
- `getSrcTablesList`
- `getTable`
- `getTargetTable`
- `getTargetColList`

Check Methods

- `getAK`
- `getAKColList`
- `getCK`
- `getColList`
- `getFK`
- `getFKColList`
- `getNotNullCol`
- `getPK`
- `getPKColList`
- `getPop`
- `getTable`
- `getTargetTable`
- `getTargetColList`

Integration Methods

- `getColList`
- `getFilter`
- `getFilterList`
- `getFrom`
- `getGrpBy`
- `getGrpByList`
- `getHaving`
- `getHavingList`
- `getJoin`
- `getJoinList`
- `getJRNFilter`
- `getJrnInfo`
- `getPop`
- `getSrcColList`
- `getSrcTablesList`
- `getTable`
- `getTargetTable`
- `getTargetColList`

Reverse Engineering Methods

- getModel

Web Services Methods

- hasPK
- nextAK
- nextCond
- nextFK

Action Methods

For more information, see Using Substitution Methods in Actions.

- getAK
- getAKColList
- getCK
- getColList
- getColumn
- getFK
- getFKColList
- getIndex
- getIndexColList
- getNewColComment
- getNewTableComment
- getPK
- getPKColList
- getTable
- getTargetTable
- isColAttrChanged

Substitution Methods Reference

getAK() method

Usage

```
public java.lang.String getAK(java.lang.String pPropertyName)
```

Description

This method returns information relative to the alternate key of a datastore during a check procedure. It is only accessible from a Check Knowledge Module if the current task is tagged "alternate key".

In an action, this method returns information related to the alternate key currently handled by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	Internal number of the AK constraint.
KEY_NAME	Name of the alternate key
MESS	Error message relative to the constraint of the alternate key
FULL_NAME	Full name of the AK generated with the local object mask.
<flexfield code>	Value of the flexfield for this AK.

Examples

The alternate key of my table is named: `<%=odiRef.getAK("KEY_NAME")%>`

getAKColList() Method

Usage

```
public java.lang.String getAKColList( java.lang.String pStart,  
java.lang.String pPattern,
```

```
java.lang.String pSeparator,  
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getAKCollList (  
java.lang.String pPattern,  
java.lang.String pSeparator)
```

Description

Offers a list of columns and expressions for the alternate key currently checked .

The pPattern parameter is interpreted and then repeated for each element of the list. It is separated from its predecessor by the pSeparator parameter. The generated string starts with pStart and ends with pEnd.

This list contains an element for each column of the current alternate key. It is accessible from a Check Knowledge Module if the current task is tagged as an "alternate key".

In an action, this method returns the list of the columns of the alternate key handled by the DDL command, ordered by their position in the key.

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameters	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of attributes that can be used in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each attribute occurrence in the pattern sequence is replaced with its value. The attributes must be between brackets. ([and])</p> <p>Example « My string [COL_NAME] is a column »</p>
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as their associated description.

Parameter value	Description
I_COL	Column internal identifier
COL_NAME	Name of the key column
COL_HEADING	Header of the key column

COL_DESC	Column description
POS	Position of the column
LONGC	Length (Precision) of the column
SCALE	Scale of the column
FILE_POS	Beginning position of the column (fixed file)
BYTES	Number of physical bytes of the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: non null)
CHECK_FLOW	Flow control flag for of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol for the column
REC_CODE_LIST	List of the record codes retained for the column
COL_NULL_IF_ERR	Processing flag for the column (0 = Reject, 1 = Set active trace to null , 2= Set inactive trace to null)
DEF_VALUE	Default value for the column
EXPRESSION	Not used
CX_COL_NAME	Not used
ALIAS_SEP	Grouping symbol used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.

<flexfield code>	Flexfield value for the current column.
------------------	---

Examples

If the CUSTOMER table has an alternate key AK_CUSTOMER (CUST_ID, CUST_NAME) and you want to generate the following code:

```
create table T_AK_CUSTOMER (CUST_ID numeric(10) not null, CUST_NAME
varchar(50) not null)
```

you just have to write:

```
create table T_<%=odiRef.getAK("KEY_NAME")%> <%=odiRef.getAKColList("(",
"[COL_NAME] [DEST_CRE_DT] not null", ", ", ", ")")%>
```

Explanation: the getAKColList function will be used to generate the part (CUST_ID numeric(10) not null, CUST_NAME varchar(50) not null), which starts and stops with a parenthesis and repeats the pattern (column, a data type, and not null) separated by commas for each column of the alternate key. Thus

- the first parameter "(" of the function indicates that we want to start the string with the string "("
- the second parameter "[COL_NAME] [DEST_CRE_DT] not null" indicates that we want to repeat this pattern for each column of the alternate key . The keywords [COL_NAME] and [DEST_CRE_DT] reference valid keywords of the Pattern Attributes List table
- the third parameter ", " indicates that we want to separate interpreted occurrences of the pattern with the string ", "
- the forth parameter ")" of the function indicates that we want to end the string with the string ")"

getCatalogName() Method

Usage

```
public java.lang.String getCatalogName(
java.lang.String pLogicalSchemaName,
java.lang.String pLocation)

public java.lang.String getCatalogName(
java.lang.String pLogicalSchemaName,
java.lang.String pContextCode,
java.lang.String pLocation)

public java.lang.String getCatalogName(
java.lang.String pLocation)

public java.lang.String getCatalogName()
```

Description

Allows you to retrieve the name of a physical data catalog or work catalog, from its logical schema.

If the first syntax is used, the returned catalog name matches the current context.

If the second syntax is used, the returned catalog name is that of the context specified in the `pContextCode` parameter.

The third syntax returns the name of the data catalog (D) or work catalog (W) for the current logical schema in the current context.

The fourth syntax returns the name of the data catalog (D) for the current logical schema in the current context.

Parameters

Parameter	Type	Description
<code>pLogicalSchemaName</code>	String	Name of the logical schema
<code>pContextCode</code>	String	Code of the enforced context of the schema
<code>pLocation</code>	String	"W" Returns the work catalog of the physical schema that corresponds to the tuple (context, logical schema)
		"D" Returns the data catalog of the physical schema corresponding to the tuple (context, logical schema)

Examples

If you have defined the physical schema:

`Pluton.db_odi.dbo`

Data catalog:	db_odi
Data schema:	dbo
Work catalog:	tempdb
Work schema:	temp_owner

that you have associated with this physical schema: MSSQL_ODI in the context CTX_DEV

The call to	Returns
<code><%=odiRef.getCatalogName("MSSQL_ODI", "CTX_DEV", "W") %></code>	tempdb
<code><%=odiRef.getCatalogName("MSSQL_ODI", "CTX_DEV", "D") %></code>	db_odi

getCatalogNameDefaultPSchema() Method

Usage

```
public java.lang.String getCatalogNameDefaultPSchema(
    java.lang.String pLogicalSchemaName,
    java.lang.String pLocation)
```

```
public java.lang.String getCatalogNameDefaultPSchema(  
    java.lang.String pLogicalSchemaName,  
    java.lang.String pContextCode,  
    java.lang.String pLocation)  
  
public java.lang.String getCatalogNameDefaultPSchema(  
    java.lang.String pLocation)  
  
public java.lang.String getCatalogNameDefaultPSchema()
```

Description

Allows you to retrieve the name of the **default** physical data catalog or work catalog for the data server to which is associated the physical schema corresponding to the tuple (logical schema, context). If no context is specified, the current context is used. If no logical schema name is specified, then the current logical schema is used. If no pLocation is specified, then the data catalog is returned.

Parameters

Parameter	Type	Description
pLogicalSchemaName	String	Name of the logical schema
pContextCode	String	Code of the enforced context of the schema
pLocation	String	"W" Returns the work catalog of the default physical schema associate to the data server to which the physical schema corresponding to the tuple (context, logical schema) is also attached.
		"D" Returns the data catalog of the physical schema corresponding to the tuple (context, logical schema)

Examples

If you have defined the physical schemas:

```
Pluton.db_odi.dbo
```

Data catalog:	db_odi
Data schema:	dbo
Work catalog:	tempdb
Work schema:	temp_odi
Default Schema	Yes

that you have associated with this physical schema: MSSQL_ODI in the context CTX_DEV, and
`Pluton.db_doc.doc`

Data catalog:	db_doc
Data schema:	doc

Work catalog:	tempdb
Work schema:	temp_doc
Default Schema	No

that you have associated with this physical schema: MSSQL_DOC in the context CTX_DEV

The call to	Returns
<code><%=odiRef.getCatalogNameDefaultPSchema("MSSQL_DOC", "CTX_DEV", "W") %></code>	tempdb
<code><%=odiRef.getCatalogNameDefaultPSchema("MSSQL_DOC", "CTX_DEV", "D") %></code>	db_odi

getCK() Method

Usage

```
public java.lang.String getCK(java.lang.String pPropertyName)
```

Description

This method returns information relative to a condition of a datastore during a check procedure. It is accessible from a Check Knowledge Module only if the current task is tagged as "condition".

In an action, this method returns information related to the check constraint currently handled by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	Current string containing the name of the requested property.

The following table lists the different values accepted by pPropertyName :

Value of the parameter	Description
ID	Internal number of the check constraint.
COND_ALIAS	Alias of the table used in the SQL statement
COND_NAME	Name of the condition
COND_TYPE	Type of the condition
COND_SQL	SQL statement of the condition
MESS	Error message relative to the check constraint
FULL_NAME	Full name of the check constraint generated with the local object mask.

COND_SQL_DDL	SQL statement of the condition with no table alias.
<flexfield code>	Flexfield value for this check constraint.

Examples

The current condition is called: `<%=snpRep.getCK("COND_NAME")%>`

```
insert into MY_ERROR_TABLE
select *
from MY_CHECKED_TABLE
where (not (<%=odiRef.getCK("COND_SQL")%>))
```

getColDefaultValue() Method

Usage

```
public java.lang.String getColDefaultValue()
```

Description

Returns the default value of the target column of the mapping.

This method can be used in a mapping expression without the `<% %>` tags. This method call will insert in the generate code the default value set in the column definition. Depending on the column type, this value should be protected with quotes.

Examples

'The default value of my target column is ' + ' `odiRef.getColDefaultValue()` '

getColList() Method

Usage

```
public java.lang.String getColList(
java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd,
java.lang.String pSelector)
```

Alternative syntaxes:

```
public java.lang.String getColList(
java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```



```
public java.lang.String getColList(  
    java.lang.String pPattern,  
    java.lang.String pSeparator,  
    java.lang.String pSelector)  
  
public java.lang.String getColList(  
    java.lang.String pPattern,  
    java.lang.String pSeparator)
```

Description

Offers a list of columns and expressions. The columns list depends on the phase during which this method is called.

The pPattern parameter is interpreted and then repeated for each element of the list (selected according to pSelector parameter) and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

In the alternative syntax, any parameters not set are set to an empty string.

Loading (LKM)

All mapping expressions that are executed in the current source environment as well as the columns used in the mapping, filters expressions, joins that are executed in the staging area.

Only the mappings tagged as "execute" in the interface are considered.

- The list is sorted on POS, FILE_POS

If there is a journalized datastore in the source of the interface, the three journalizing pseudo columns JRN_FLG, JRN_DATE, and JRN_SUBSCRIBER are added as columns of the journalized source datastore.

Integration (IKM)

All current mapping expressions tagged as "execute" in the current interface.

The list contains one element for each column that is loaded (tagged as "execute") in the target table of the current interface.

- The list is sorted on POS, FILE_POS when the table loaded is not temporary.
- The list is not sorted when the table loaded is a temporary table (does not exist in the referential).

If there is a journalized datastore in the source of the interface, and it is located in the staging area, the three journalizing pseudo columns JRN_FLG, JRN_DATE, and JRN_SUBSCRIBER are added as columns of the journalized source datastore.

Check (CKM)

All the columns of the **target** table (with static or flow control)

To distinguish the columns of the target table from those filled in the current interface, you must use the MAP selector.

- The list is sorted on POS, FILE_POS of the target table

Actions

All the columns of the table handles by the DDL command.

In the case of modified, added or deleted columns, the NEW and OLD selectors are used to retrieve either the new version of the old version of the modified column being processed by the DDL command.

- The list is sorted on POS, FILE_POS of the table

Parameters

Parameter	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of the attributes usable in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each occurrence of the attributes in the pattern string is replaced by its value. Attributes must be between brackets ([and])</p> <p>Example « My string [COL_NAME] is a column »</p>
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.
pSelector	String	<p>String that designates a Boolean expression that allows to filter the elements of the initial list with the following format :</p> <p><SELECTOR> <Operator> <SELECTOR> etc. Parenthesis are authorized.</p> <p>Authorized operators:</p> <ol style="list-style-type: none">1. No: NOT or !2. Or: OR or 3. And: AND or && <p>Example: (INS AND UPD) OR TRG</p> <p>The description of valid selectors is provided in the table « Selectors Description »</p>

Pattern Attributes List

The following table lists different parameters values as well as their associated description.

Parameter value	Description
I_COL	Internal identifier of the column
COL_NAME	Name of the column
COL_HEADING	Header of the column
COL_DESC	Description of the column
POS	Position of the column

LONGC	Column length (Precision)
SCALE	Scale of the column
FILE_POS	Beginning (index) of the column
BYTES	Number of physical bytes in the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: not null)
CHECK_FLOW	Flow control flag of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol of the column
REC_CODE_LIST	List of the record codes retained in the column
COL_NULL_IF_ERR	Processing flag of the column (0 = Reject, 1 = Set to null active trace, 2= set to null inactive trace)
DEF_VALUE	Default value of the column
EXPRESSION	Text of the expression executed on the source (expression as typed in the mapping or column name making an expression executed on the staging area).
CX_COL_NAME	Computed name of the column used as a container for the current expression on the staging area
ALIAS_SEP	Separator used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
MANDATORY_CLAUSE	Returns NOT NULL if the column is mandatory. Otherwise, returns the

	null keyword for the technology.
DEFAULT_CLAUSE	Returns <code>DEFAULT <default value></code> if any default value exists. Otherwise, returns an empty string.
COL_DESC	Description (comment) of the column. Quotes and double quotes are replaced with spaces.
JDBC_TYPE	Data Services - JDBC Type of the column returned by the driver.
<flexfield code>	Flexfield value for the current column.

Selectors Description

Parameters value	Description
INS	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: Only for mapping expressions marked with insertion• CKM: Not applicable
UPD	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: Only for the mapping expressions marked with update• CKM: Non applicable
TRG	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: Only for the mapping expressions executed on the target• CKM: Mapping expressions executed on the target.
NULL	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: All mapping expressions loading not nullable columns• CKM: All target columns that do not accept null values
PK	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: All mapping expressions loading the primary key columns• CKM: All the target columns that are part of the primary key
UK	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: All the mapping expressions loading the update key column chosen for the current interface.• CKM: Not applicable.
REW	<ul style="list-style-type: none">• LKM: Not applicable (*)• IKM: All the mapping expressions loading the columns with read only flag not selected.• CKM: All the target columns with read only flag not selected.
UD1	<ul style="list-style-type: none">• LKM: Not applicable (*)

	<ul style="list-style-type: none"> • IKM: All mapping expressions loading the columns marked UD1 • CKM: Not applicable
UD2	<ul style="list-style-type: none"> • LKM: Not applicable (*) • IKM: All mapping expressions loading the columns marked UD2 • CKM: Not applicable
UD3	<ul style="list-style-type: none"> • LKM: Not applicable (*) • IKM: All mapping expressions loading the columns marked UD3 • CKM: Not applicable
UD4	<ul style="list-style-type: none"> • LKM: Not applicable (*) • IKM: All mapping expressions loading the columns marked UD4 • CKM: Not applicable
UD5	<ul style="list-style-type: none"> • LKM: Not applicable (*) • IKM: All mapping expressions loading the columns marked UD5 • CKM: Not applicable
MAP	<ul style="list-style-type: none"> • LKM: Not applicable • IKM: Not applicable • CKM: <p>Flow control: All columns of the target table loaded with expressions in the current interface</p> <p>Static control: All columns of the target table</p>
SCD_SK	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Surrogate Key in the data model definition.
SCD_NK	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Natural Key in the data model definition.
SCD_UPD	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Overwrite on Change in the data model definition.
SCD_INS	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Add Row on Change in the data model definition.
SCD_FLAG	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Current Record Flag in the data model definition.
SCD_START	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Starting Timestamp in the data model definition.
SCD_END	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Ending Timestamp in the data model definition.
NEW	<ul style="list-style-type: none"> • Actions: the column added to a table, the new version of the modified column of a table.

OLD	<ul style="list-style-type: none">• Actions: The column dropped from a table, the old version of the modified column of a table.
WS_INS	<ul style="list-style-type: none">• SKM: The column is flagged as allowing INSERT using Data Services.
WS_UPD	<ul style="list-style-type: none">• SKM: The column is flagged as allowing UPDATE using Data Services.
WS_SEL	<ul style="list-style-type: none">• SKM: The column is flagged as allowing SELECT using Data Services.

(*) Important Note : Using certain selectors in a LKM - indicated with a * - is possible but not recommended. Only columns mapped on the source in the interface are returned. **As a consequence, the result could be incorrect depending on the interface.**

For example, for the UK selector, the columns of the key that are not mapped or that are not executed on the source will not be returned with the selector.

Examples

If the CUSTOMER table contains the columns (CUST_ID, CUST_NAME, AGE) and we want to generate the following code:

```
create table CUSTOMER (CUST_ID numeric(10) null, CUST_NAME varchar(50)
null, AGE numeric(3) null)
```

we just need to write:

```
create table CUSTOMER <%=odiRef.getColList("(", "[COL_NAME]
[SOURCE_CRE_DT] null", ", ", ") ", "")%>
```

Explanation: the getColList function will be used to generate (CUST_ID numeric(10) null, CUST_NAME varchar(50) null, AGE numeric(3) null). It will start and end with a parenthesis and repeat a pattern (column, data type, and null) separated by commas for each column. Thus,

- the first character "(" of the function indicates that we want to start the string with the string "("
- the second parameter "[COL_NAME] [SOURCE_CRE_DT] null" indicates that we want to repeat this pattern for each column. The keywords [COL_NAME] and [SOURCE_CRE_DT] are references to valid keywords of the table Pattern Attribute List
- the third parameter ", " indicates that we want to separate the interpreted occurrences of the pattern with the string ", ".
- the fourth parameter ")" of the function indicates that we want to end the string with the string ")"
- the last parameter "" indicates that we want to repeat the pattern for each column (with no selection)

getColumn() Method

Usage

```
public java.lang.String getColumn(
java.lang.String pPattern,
java.lang.String pSelector)

public java.lang.String getColumn(
java.lang.String pPattern)
```

Description

In an action, returns information on a column being handled by an the action.

Parameters

Parameter	Type	Description
pPattern	String	<p>Pattern of values rendered for the column.</p> <p>The list of the attributes usable in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each occurrence of the attributes in the pattern string is replaced by its value. Attributes must be between brackets ([and])</p> <p>Example « My string [COL_NAME] is a column »</p>
pSelector	String	<p>The Selector may take one of the following value:</p> <ul style="list-style-type: none"> NEW: returns the new version of the modified column or the new column. OLD: returns the old version of the modified column or the dropped column. <p>If the selector is omitted, it is set to OLD for all <i>drop</i> actions. Otherwise, it is set to NEW.</p>

Pattern Attributes List

The following table lists different parameters values as well as their associated description.

Parameter value	Description
I_COL	Internal identifier of the column
COL_NAME	Name of the column
COL_HEADING	Header of the column
COL_DESC	Description of the column
POS	Position of the column
LONGC	Column length (Precision)
SCALE	Scale of the column
FILE_POS	Beginning (index) of the column
BYTES	Number of physical bytes in the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: not null)

CHECK_FLOW	Flow control flag of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol of the column
REC_CODE_LIST	List of the record codes retained in the column
COL_NULL_IF_ERR	Processing flag of the column (0 = Reject, 1 = Set to null active trace, 2= set to null inactive trace)
DEF_VALUE	Default value of the column
EXPRESSION	Text of the expression executed on the source (expression as typed in the mapping or column name making an expression executed on the staging area).
CX_COL_NAME	Computed name of the column used as a container for the current expression on the staging area
ALIAS_SEP	Separator used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
MANDATORY_CLAUSE	Returns <code>NOT NULL</code> if the column is mandatory. Otherwise, returns the null keyword for the technology.
DEFAULT_CLAUSE	Returns <code>DEFAULT <default value></code> if any default value exists. Otherwise, returns an empty string.
COL_DESC	Description (comment) of the column. Quotes and double quotes are replaced with spaces.
<flexfield code>	Flexfield value for the current column.

getContext() Method

Usage

```
public java.lang.String getContext(java.lang.String pPropertyName)
```

Description

This method returns information about to the current execution context.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	Internal ID of the context.
CTX_NAME	Name of the context.
CTX_CODE	Code of the context.
CTX_DEFAULT	Returns 1 for the default context, 0 for the other contexts.
<flexfield code>	Flexfield value for this reference.

Examples

```
Current Context = <%=getContext("CTX_NAME")%>
```

getDataType() Method

Usage

```
public java.lang.String getDataType(  
java.lang.String pDataTypeName,  
java.lang.String pDataTypeLength,  
java.lang.String pDataTypePrecision)
```

Description

Returns the creation syntax of the following SQL data types: varchar, numeric or date according to the parameters associated to the source or target technology.

Parameters

Parameter	Type	Description
-----------	------	-------------

pDataTypeName	String	Name of the data type as listed in the table below
pDataTypeLength	String	Length of the data type
pDataTypePrecision	String	Precision of the data type

The following table lists all possible values for pDataTypeName

Value	Description
SRC_VARCHAR	Returns the syntax to the source data type varchar
SRC_NUMERIC	Returns the syntax to the source data type numeric
SRC_DATE	Returns the syntax to the source data type date
DEST_VARCHAR	Returns the syntax to the target data type varchar
DEST_NUMERIC	Returns the syntax to the target data type numeric
DEST_DATE	Returns the syntax to the target data type date

Examples

Given the following syntax for these technologies:

Technology	Varchar	Numeric	Date
Oracle	varchar2(%L)	number(%L,%P)	date
MS SqlServer	varchar(%L)	numeric(%L,%P)	datetime
MS Access	Text(%L)	double	datetime

Here are some examples of call to getDataType:

Call	Oracle	MS SqlServer	MS Access
<code><%=odiRef.getDataType("DEST_VARCHAR", "10", "")%></code>	varchar2(10)	varchar(10)	Text(10)
<code><%=odiRef.getDataType("DEST_VARCHAR", "10", "5")%></code>	varchar2(10)	varchar(10)	Text(10)
<code><%=odiRef.getDataType("DEST_NUMERIC", "10", "")%></code>	number(10)	numeric(10)	double
<code><%=odiRef.getDataType("DEST_NUMERIC", "10", "2")%></code>	number(10,2)	numeric(10,2)	double
<code><%=odiRef.getDataType("DEST_NUMERIC", "", "")%></code>	number	numeric	double
<code><%=odiRef.getDataType("DEST_DATE", "", "")%></code>	date	datetime	datetime

```
<%=odiRef.getDataType("DEST_DATE", "10", date datetime datetime
"2") %>
```

getFilter() Method

Usage

```
public java.lang.String getFilter()
```

Description

Returns the SQL filters sequence (on the source while loading, on the staging area while integrating).

Parameters

None

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W") %>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "INS=1") %>
from <%=odiRef.getFrom() %>
where (1=1)
<%=odiRef.getJoin() %>
<%=odiRef.getFilter() %>
<%=odiRef.getGrpBy() %>
<%=odiRef.getHaving() %>
```

getFilterList() Method

Usage

```
public java.lang.String getFilterList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getFilterList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Offers a list of occurrences of the SQL filters of an interface.

The parameter pPattern is interpreted and repeated for each element of the list and separated from its predecessor with parameter pSeparator. The generated string begins with pStart and ends with pEnd.

This list contains an element for each filter expression executed on the source or target (depending on the Knowledge Module in use).

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
		The pattern will be repeated for each occurrence of the list.
		The list of possible in a list is available in the table « Pattern Attributes List »
pPattern	String	Each attribute occurrence in the pattern string is substituted with its value. Attributes must be between brackets ([and]) Example « My string [COL_NAME] is a column »
pSeparator	String	This parameter is used to separate a pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as the associated description.

Parameter value	Description
ID	Filter internal identifier
EXPRESSION	Text of the filter expression.

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
<%=odiRef.getJoin()%>
<%=odiRef.getFilterList("and ", "([EXPRESSION])", " and ", "")%>
<%=odiRef.getGrpBy()%>
<%=odiRef.getHaving()%>
```

Explanation: the getFilterList function will be used to generate the filter of the SELECT clause that must begin with "and" and repeats the pattern (expression of each filter) separated with "and" for each filter. Thus

- The first parameter **"and"** of the function indicates that we want to start the string with the string "and"
- the second parameter **"([EXPRESSION])"** indicates that we want to repeat this pattern for each filter. The keywords [EXPRESSION] references a valid keyword of the table Pattern Attribute List
- the third parameter **" and "** indicates that we want to separate each interpreted occurrence of the pattern with the string "and ".

- the fourth parameter "" of the function indicates that we want to end the string with no specific character.

getFK() Method

Usage

```
public java.lang.String getFK(java.lang.String pPropertyName)
```

Description

This method returns information relative to the foreign key (or join or reference) of a datastore during a check procedure. It is accessible from a Knowledge Module only if the current task is tagged as a "reference".

In an action, this method returns information related to the foreign key currently handled by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	Internal number of the reference constraint.
FK_NAME	Name of the reference constraint.
FK_TYPE	Type of the reference constraint.
FK_ALIAS	Alias of the reference table (only used in case of a complex expression)
PK_ALIAS	Alias of the referenced table (only used in case of a complex expression)
ID_TABLE_PK	Internal number of the referenced table.
PK_I_MOD	Number of the referenced model.
PK_CATALOG	Catalog of the referenced table.
PK_SCHEMA	Physical schema of the referenced table.
PK_TABLE_NAME	Name of the referenced table.
COMPLEX_SQL	Complex SQL statement of the join clause (if appropriate).
MESS	Error message of the reference constraint
FULL_NAME	Full name of the foreign key generated with the local object mask.

<flexfield code> Flexfield value for this reference.

Examples

The current reference key of my table is called: `<%=odiRef.getFK("FK_NAME")%>`. It references the table `<%=odiRef.getFK("PK_TABLE_NAME")%>` that is in the schema `<%=odiRef.getFK("PK_SCHEMA")%>`

getFKColList() Method

Usage

```
public java.lang.String getFKColList( java.lang.String pStart,  
java.lang.String pPattern,  
java.lang.String pSeparator,  
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getFKColList(  
java.lang.String pPattern,  
java.lang.String pSeparator)
```

Description

Offers a list of columns part of a reference constraint (foreign key).

The parameter pPattern is interpreted and repeated for each element of the list, and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

This list contains one element for each column of the current foreign key. It is accessible from a Check Knowledge Module only if the current task is tagged as a "reference".

In an action, this method returns the list of the columns of the foreign key handled by the DDL command, ordered by their position in the key.

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This parameter marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of possible attributes in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each attribute occurrence in the pattern string is substituted with its value. The attributes must be between brackets ([and])</p> <p>Example « My string [COL_NAME] is a column »</p>
pSeparator	String	This parameter separates each pattern from its predecessor.

pEnd String This parameter marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as the associated description.

Parameter value	Description
I_COL	Column internal identifier
COL_NAME	Name of the column of the key
COL_HEADING	Header of the column of the key
COL_DESC	Description of the column of the key
POS	Position of the column of the key
LONGC	Length (Precision) of the column of the key
SCALE	Scale of the column of the key
FILE_POS	Beginning (index) of the column
BYTES	Number of physical octets of the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: not authorized, 1: not null)
CHECK_FLOW	Flow control flag of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol for the column
REC_CODE_LIST	List of the record codes for the column
COL_NULL_IF_ERR	Column processing flag (0 = Reject, 1 = set active trace to null, 2= set inactive trace to null)
DEF_VALUE	Default value of the column
EXPRESSION	Not used
CX_COL_NAME	Not used
ALIAS_SEP	Separator used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.

SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
PK_I_COL	Internal identifier of the referenced column
PK_COL_NAME	Name of the referenced key column
PK_COL_HEADING	Header of the referenced key column
PK_COL_DESC	Description of the referenced key column
PK_POS	Position of the referenced column
PK_LONGC	Length of the referenced column
PK_SCALE	Precision of the referenced column
PK_FILE_POS	Beginning (index) of the referenced column
PK_BYTES	Number of physical octets of the referenced column
PK_FILE_END_POS	End of the referenced column (FILE_POS + BYTES)
PK_IND_WRITE	Write right flag of the referenced column
PK_COL_MANDATORY	Mandatory character of the referenced column(0: null authorized, 1: not null)
PK_CHECK_FLOW	Flow control flag of the referenced column (0: do not check, 1: check)
PK_CHECK_STAT	Static control flag of the referenced column (0: do not check, 1: check)
PK_COL_FORMAT	Logical format of the referenced column
PK_COL_DEC_SEP	Decimal separator for the referenced column
PK_REC_CODE_LIST	List of record codes retained for the referenced column
PK_COL_NULL_IF_ERR	Processing flag of the referenced column (0 = Reject, 1 = Set active trace to null, 2= Set inactive trace to null)
PK_DEF_VALUE	Default value of the referenced column
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
<flexfield code>	Flexfield value for the current column of the referencing table.

Examples

If the CUSTOMER table references the CITY table on CUSTOMER.COUNTRY_ID = CITY.ID_COUNT and CUSTOMER.CITY_ID = CITY.ID_CIT

the clause:

```
(CUS.COUNTRY_ID = CITY.ID_COUNT and CUS.CITY_ID = CITY.ID_CIT)
```

can also be written:

```
<%=odiRef.getFKColList("(" , "CUS.[COL_NAME] = CITY.[PK_COL_NAME]", " and", " )" )%>
```

Explanation: the getFKColList function will be used to loop on each column of the foreign key to generate the clause that begins and ends with a parenthesis and that repeats a pattern separated by **and** for each column in the foreign key. Thus

- The first parameter "(" of the function indicates that we want to begin the string with " ("
- The second parameter "CUS.[COL_NAME] = CITY.[PK_COL_NAME]" indicates that we want to repeat this pattern for each column of the foreign key. The keywords [COL_NAME] and [PK_COL_NAME] reference valid keywords in the table Pattern Attributes List
- The third parameter " and " indicates that we want to separate the occurrences of the pattern with the string " and ".
- The fourth parameter ")" of the function indicates that we want to end the string with ") ".

getFlexFieldValue Method

Usage

```
public java.lang.String getFlexFieldValue(java.lang.String pI_Instance,
java.lang.String pI_Object, java.lang.String pFlexFieldCode)
```

Description

This method returns the value of an Object Instance's Flexfield.

Parameters

Parameter	Type	Description
pI_Instance	String	Internal Identifier of the Object Instance, as it appears in the version tab of the object instance window.
pI_Object	String	Internal Identifier of the Object type, as it appears in the version tab of the object window for the object type.
pPropertyName	String	Flexfield Code which value should be returned.

Examples

```
<%=odiRef.getFlexFieldValue("32001","2400","MY_DATASTORE_FIELD")%>
```

Returns the value of the flexfield `MY_DATASTORE_FIELD`, for the object instance of type datastore (Internal ID for datastores is 2400), with the internal ID 32001.

getFrom() Method

Usage

```
public java.lang.String getFrom()
```

Description

Allows the retrieval of the SQL string of the **FROM** in the source **SELECT** clause. The **FROM** statement is built from tables and joins (and according to the SQL capabilities of the technologies) that are used in an interface. Thus, for a technology that supports ISO outer joins and parenthesis, `getFrom()` could return a string such as:

```
((CUSTOMER as CUS inner join CITY as CIT on (CUS.CITY_ID = CIT.CITY_ID))  
left outer join SALES_PERSON as SP on (CUS.SALES_ID = SP.SALE_ID))
```

If there is a journalized datastore in source of the interface, the source table in the clause is replaced by the data view linked to the journalized source datastore.

Parameters

None

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>  
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>  
from <%=odiRef.getFrom()%>  
where (1=1)  
<%=odiRef.getJoin()%>  
<%=odiRef.getFilter()%>  
<%=odiRef.getGrpBy()%>  
<%=odiRef.getHaving()%>
```

getGrpBy() Method

Usage

```
public java.lang.String getGrpBy()
```

Description

Allows you to retrieve the SQL GROUP BY string (on the "source" during the collect phase, on the staging area during the integration phase). This statement is automatically computed from the aggregation transformations detected in the mapping expressions.

Parameters

None

Example

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
<%=odiRef.getJoin()%>
<%=odiRef.getFilter()%>
<%=odiRef.getGrpBy()%>
<%=odiRef.getHaving()%>
```

getGrpByList() Method

Usage

```
public java.lang.String getGrpByList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getGrpByList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Offers a list of occurrences of SQL GROUP BY of an interface.

The pPattern parameter is interpreted, then repeated for each element of the list and separated from its predecessor with the pSeparator parameter. The generated string begins with pStart and ends with pEnd.

This list contains an element for each GROUP BY statement on the source or target (according to the Knowledge Module that used it).

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This parameter marks the beginning of the string to generate.
		The pattern is repeated for each occurrence in the list.
		The list of possible attributes in a pattern is detailed in the table « Pattern Attributes List »
pMotif	String	Each attribute occurrence in the pattern string is substituted with its value. The attributes must be between brackets ([and])

Example « My string [COL_NAME] is a column »

pSeparator String This parameter is used to separate each pattern from its predecessor.

pEnd String This parameter marks the end of the string to be generated.

Pattern Attributes List

The following table lists the different values of the parameters as well as their associated description.

Parameter value	Description
ID	Internal identifier of the clause
EXPRESSION	Text of the grouping statement

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=getColList("", "[EXPRESSION]", " , ", " ", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
<%=odiRef.getJoin()%>
<%=odiRef.getFilter()%>
<%=odiRef.getGrpByList("group by ", "[EXPRESSION]", " , ", " ")%>
<%=odiRef.getHaving()%>
```

Explanation: the `getGrpByList` function will be used to generate the **group by** clause of the **select** order that must start with "group by" and that repeats a pattern (each grouping expression) separated by commas for each expression.

- The first parameter "**group by**" of the function indicates that we want to start the string with "group by"
- The second parameter "**[EXPRESSION]**" indicates that we want to repeat this pattern for each group by expression. The keyword `[EXPRESSION]` references a valid keyword of the table Pattern Attributes List
- The third parameter " , " indicates that we want to separate the interpreted occurrences of the pattern with a comma.
- The fourth parameter " " of the function indicates that we want to end the string with no specific character

getHaving() Method

Usage

```
public java.lang.String getHaving()
```

Description

Allows the retrieval of the SQL statement HAVING (on the source during loading, on the staging area during integration). This statement is automatically computed from the filter expressions containing detected aggregation functions.

Parameters

None

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
<%=odiRef.getJoin()%>
<%=odiRef.getFilter()%>
<%=odiRef.getGrpBy()%>
<%=odiRef.getHaving()%>
```

getHavingList() Method

Usage

```
public java.lang.String getHavingList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getHavingList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Offers a list of the occurrences of SQL HAVING of an interface.

The parameter pPattern is interpreted and repeated for each element of the list, and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

This list contains one element for each HAVING expression to execute on the source or target (depends on the Knowledge module that uses it).

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This parameter marks the beginning of the string to generate.
pPattern	String	The pattern is repeated for each occurrence in the list. The list of authorized attributes in a pattern is detailed in the table « Pattern

Attributes List »

Each attribute occurrence in the pattern string is substituted with its value.
The attributes must be between brackets ([and])

Example « My string [COL_NAME] is a column »

pSeparator String This parameter separates each pattern from its predecessor.

pEnd String This parameter marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as the associated description.

Parameter value	Description
ID	Internal identifier of the clause
EXPRESSION	Text of the having expression

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
<%=odiRef.getJoin()%>
<%=odiRef.getFilter()%>
<%=odiRef.getGrpByList("group by ", "[EXPRESSION]", " , ", "")%>
<%=odiRef.getHavingList("having ", "([EXPRESSION])", " and ", "")%>
```

Explanation: The getHavingList function will be used to generate the having clause of the select order that must start with "having" and that repeats a pattern (each aggregated filtered expression) separated by "and" for each expression.

- The first parameter **"having"** of the function indicates that we want to start the string with "having"
- The second parameter **"([EXPRESSION])"** indicates that we want to repeat this pattern for each aggregated filter. The keyword [EXPRESSION] references a valid keyword of the table Pattern Attributes List
- The third parameter **"and"** indicates that we want to separate each interpreted occurrence of the pattern with the string " and ".
- The fourth parameter **""** of the function indicates that we want to end the string with no specific character

getIndex() Method

Usage

```
public java.lang.String getIndex(java.lang.String pPropertyName)
```

Description

In an action, this method returns information related to the index currently handled by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	Internal number of the index.
KEY_NAME	Name of the index
FULL_NAME	Full name of the index generated with the local object mask.
<flexfield code>	Value of the flexfield for this index.

getIndexColList() Method

Usage

```
public java.lang.String getIndexColList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Description

In an action, this method returns the list of the columns of the index handled by the DDL command, ordered by their position in the index.

The pPattern parameter is interpreted and then repeated for each element of the list. It is separated from its predecessor by the pSeparator parameter. The generated string starts with pStart and ends with pEnd.

This list contains an element for each column of the current index.

Parameters

Parameters	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	The pattern is repeated for each occurrence in the list. The list of attributes that can be used in a pattern is detailed in the table

« Pattern Attributes List »

Each attribute occurrence in the pattern sequence is replaced with its value.
The attributes must be between brackets. ([and])

Example « My string [COL_NAME] is a column »

pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as their associated description.

Parameter value	Description
I_COL	Column internal identifier
COL_NAME	Name of the index column
COL_HEADING	Header of the index column
COL_DESC	Column description
POS	Position of the column
LONGC	Length (Precision) of the column
SCALE	Scale of the column
FILE_POS	Beginning position of the column (fixed file)
BYTES	Number of physical bytes of the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the columnn (0: null authorized, 1: non null)
CHECK_FLOW	Flow control flag for of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol for the column
REC_CODE_LIST	List of the record codes retained for the column
COL_NULL_IF_ERR	Processing flag for the column (0 = Reject, 1 = Set active trace to null , 2= Set inactive trace to null)
DEF_VALUE	Default value for the column

EXPRESSION	Not used
CX_COL_NAME	Not used
ALIAS_SEP	Grouping symbol used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
<flexfield code>	Flexfield value for the current column.

getInfo() Method

Usage

```
public java.lang.String getInfo(java.lang.String pPropertyName)
```

Description

Generic method that returns generic information about the current task. The list of available information is described in the **pPropertyName values** table.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

pPropertyName values

The following table lists the different values possible for pPropertyName :

Parameter value	Description
I_SRC_SET	Internal identifier of the current Source Set if the task belongs to a Loading Knowledge Module

SRC_SET_NAME	Name of the current Source Set if the task belongs to a Loading Knowledge Module
COLL_NAME	Name of the current loading resource (C\$) if the task belongs to a Loading Knowledge Module
INT_NAME	Name of the current integration resource (I\$) if the task belongs to a string Loading, Integration or Check Knowledge Module.
ERR_NAME	Name of the current error resource (E\$) if the task is part of a Loading, Integration or Check Knowledge Module
TARG_NAME	Name of the target resource if the task is part of a Loading, Integration or Check Knowledge Module
SRC_CATALOG	Name of the data catalog in the source environment
SRC_SCHEMA	Name of the data schema in the source environment
SRC_WORK_CATALOG	Name of the work catalog in the source environment
SRC_WORK_SCHEMA	Name of the work schema in the source environment
DEST_CATALOG	Name of the data catalog in the target environment
DEST_SCHEMA	Name of the data schema in the target environment
DEST_WORK_CATALOG	Name of the work catalog in the target environment
DEST_WORK_SCHEMA	Name of the work schema in the target environment
SRC_TECHNO_NAME	Name of the source technology
SRC_CON_NAME	Name of the source connection
SRC_DSERV_NAME	Name of the data server of the source machine
SRC_CONNECT_TYPE	Connection type of the source machine
SRC_IND_JNDI	JNDI URL flag
SRC_JAVA_DRIVER	Name of the JDBC driver of the source connection
SRC_JAVA_URL	JDBC URL of the source connection
SRC_JNDI_AUTHENT	JNDI authentication type
SRC_JNDI_PROTO	JNDI source protocol
SRC_JNDI_FACTORY	JNDI source Factory
SRC_JNDI_URL	Source JNDI URL
SRC_JNDI_RESSOURCE	Accessed source JNDI resource
SRC_USER_NAME	User name of the source connection

SRC_ENCODED_PASS	Encrypted password of the source connection
SRC_FETCH_ARRAY	Size of the source array fetch
SRC_BATCH_UPDATE	Size of the source batch update
SRC_EXE_CHANNEL	Execution canal of the source connection
SRC_COL_ALIAS_WORD	Term used to separated the columns from their aliases for the source technology
SRC_TAB_ALIAS_WORD	Term used to separated the tables from their aliases for the source technology
SRC_DATE_FCT	Function returning the current date for the source technology
SRC_DDL_NULL	Returns the definition used for the keyword NULL during the creation of a table on the source
SRC_MAX_COL_NAME_LEN	Maximum number of characters for the column name on the source technology
SRC_MAX_TAB_NAME_LEN	Maximum number of characters for the table name on the source technology
SRC_REM_OBJ_PATTERN	Substitution model for a remote object on the source technology.
SRC_LOC_OBJ_PATTERN	Substitution model for a local object name on the source technology.
DEST_TECHNO_NAME	Name of the target technology
DEST_CON_NAME	Name of the target connection
DEST_DSERV_NAME	Name of the data server of the target machine
DEST_CONNECT_TYPE	Connection type of the target machine
DEST_IND_JNDI	Target JNDI URL flag
DEST_JAVA_DRIVER	Name of the JDBC driver of the target connection
DEST_JAVA_URL	JDBC URL of the target connection
DEST_JNDI_AUTHENT	JNDI authentication type of the target
DEST_JNDI_PROTO	JNDI target protocol
DEST_JNDI_FACTORY	JNDI target Factory
DEST_JNDI_URL	JNDI URL of the target
DEST_JNDI_RESSOURCE	Target JNDI resource that is accessed
DEST_USER_NAME	Name of the user for the target connection
DEST_ENCODED_PASS	Encrypted password for the target connection

DEST_FETCH_ARRAY	Size of the target array fetch
DEST_BATCH_UPDATE	Size of the target batch update
DEST_EXE_CHANNEL	Execution canal of the target connection
DEST_COL_ALIAS_WORD	Term used to separate the columns from their aliases on the target technology
DEST_TAB_ALIAS_WORD	Term used to separate the tables from their aliases on the target technology
DEST_DATE_FCT	Function returning the current date on the target technology
DEST_DDL_NULL	Function returning the definition used for the keyword NULL during the creation on a table on the target
DEST_MAX_COL_NAME_LEN	Maximum number of characters of the column in the target technology
DEST_MAX_TAB_NAME_LEN	Maximum number of characters of the table name on the target technology
DEST_REM_OBJ_PATTERN	Substitution model for a remote object on the target technology
DEST_LOC_OBJ_PATTERN	Substitution model for a local object name on the target technology
CT_ERR_TYPE	Error type (F: Flow, S: Static). Applies only in the case of a Check Knowledge Module
CT_ERR_ID	Error identifier (Table # for a static control or interface number for flow control. Applies only in the case of a Check Knowledge Module
CT_ORIGIN	Name that identifies the origin of an error (Name of a table for static control, or name of an interface prefixed with the project code). Applies only in the case of a Check Knowledge Module
JRN_NAME	Name of the journalized datastore.
JRN_VIEW	Name of the view linked to the journalized datastore.
JRN_DATA_VIEW	Name of the data view linked to the journalized datastore.
JRN_TRIGGER	Name of the trigger linked to the journalized datastore.
JRN_ITRIGGER	Name of the Insert trigger linked to the journalized datastore.
JRN_UTRIGGER	Name of the Update trigger linked to the journalized datastore.
JRN_DTRIGGER	Name of the Delete trigger linked to the journalized datastore.
SUBSCRIBER_TABLE	Name of the datastore containing the subscribers list.
CDC_SET_TABLE	Full name of the table containing list of CDC sets.

CDC_TABLE_TABLE	Full name of the table containing the list of tables journalized through CDC sets.
CDC_SUBS_TABLE	Full name of the table containing the list of subscribers to CDC sets.
CDC_OBJECTS_TABLE	Full name of the table containing the journalizing parameters and objects.
SRC_DEF_CATALOG	Default catalog for the source data server.
SRC_DEF_SCHEMA	Default schema for the source data server.
SRC_DEFW_CATALOG	Default work catalog for the source data server.
SRC_DEFW_SCHEMA	Default work schema for the source data server.
DEST_DEF_CATALOG	Default catalog for the target data server.
DEST_DEF_SCHEMA	Default schema for the target data server.
DEST_DEFW_CATALOG	Default work catalog for the target data server.
DEST_DEFW_SCHEMA	Default work schema for the target data server.
SRC_LSCHEMA_NAME	Source logical schema name.
DEST_LSCHEMA_NAME	Target logical schema name.

Examples

The current source connection is: `<%=odiRef.getInfo("SRC_CON_NAME")%>`
on server: `<%=odiRef.getInfo("SRC_DSERV_NAME")%>`

getJDBCConnection() method

Usage

```
java.sql.Connection getJDBCConnection(
java.lang.String pPropertyName)
```

Description

This method returns the source or target JDBC connection for the current task.

Warning! This method does not return a string, but a JDBC connection object. This object may be used in your Java code within the task.

Parameters

Parameter	Type	Description
-----------	------	-------------

pPropertyName	String	Name of connection to be returned.
---------------	--------	------------------------------------

pPropertyName values

The following table lists the different values possible for pPropertyName :

Parameter value	Description
SRC	Source connection for the current task.
DEST	Target connection for the current task.

Examples

Gets the source connection and creates a statement for this connection.

```
java.sql.Connection sourceConnection = odiRef.getJDBCConnection("SRC");  
java.sql.Statement s = sourceConnection.createStatement();
```

getJoin() Method

Usage

```
public java.lang.String getJoin()
```

Description

Retrieves the SQL join string (on the source during the loading, on the staging area during the integration).

Parameters

None

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>  
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>  
from <%=odiRef.getFrom()%>  
where (1=1)  
<%=odiRef.getJoin()%>  
<%=odiRef.getFilter()%>  
<%=odiRef.getGrpBy()%>  
<%=odiRef.getHaving()%>
```

getJoinList() Method

Usage

```
public java.lang.String getJoinList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getJoinList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Offers a list of the occurrences of the SQL joins in an interface to position in a WHERE clause.

The pPattern parameter is interpreted and then repeated for each element in the list and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends up with pEnd.

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This parameter marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of authorized attributes in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each attribute occurrence in the pattern string is substituted with its value. The attributes must be between brackets ([and])</p> <p>Example My string [COL_NAME] is a column »</p>
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This parameter marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as the associated description.

Parameter value	Description
ID	Internal identifier of the join
EXPRESSION	Text of the join expression

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "", "INS=1")%>
from <%=odiRef.getFrom()%>
where (1=1)
```

```
<%=odiRef.getJoinList("and ", "([EXPRESSION]) ", " and ", "")%>  
<%=odiRef.getFilterList("and ", "([EXPRESSION]) ", " and ", "")%>  
<%=odiRef.getGrpBy()%>  
<%=odiRef.getHaving()%>
```

Explanation: the `getJoinList` function will be used to generate join expressions to put in the WHERE part of the SELECT statement that must start with "and" and that repeats a pattern (the expression of each join) separated by " and " for each join. Thus :

- The first parameter **"and"** of the function indicates that we want to start the string with "and"
- The second parameter **"([EXPRESSION])"** indicates that we want to repeat this pattern for each join. The keyword [EXPRESSION] references a valid keyword of the table Pattern Attributes List
- The third parameter **" and "** indicates that we want to separate each interpreted occurrence of the pattern with " and " (note the spaces before and after "and")
- The fourth parameter **""** of the function indicates that we want to end the string with no specific character

getJrnFilter() Method

Usage

```
public java.lang.String getJrnFilter()
```

Description

Returns the SQL Journalizing filter for the current interface. If the journalized table in the source, this method can be used during the loading phase. If the journalized table in the staging area, this method can be used while integrating.

Parameters

None

Examples

```
<%=odiRef.getJrnFilter()%>
```

getJrnInfo() Method

Usage

```
public java.lang.String getJrnInfo(java.lang.String pPropertyName)
```

Description

Returns generic information about a datastore's journalizing for a JKM while journalizing a model/datastore, or for a LKM/IKM in an interface.

Parameters

Parameter	Type	Description
pPropertyName	String	String containing the name of the requested property.

pPropertyName values

The following table lists the different values possible for pPropertyName :

Parameter value	Description
FULL_TABLE_NAME	Full name of the journalized datastore.
JRN_FULL_NAME	Full name of the journal datastore.
JRN_FULL_VIEW	Full name of the view linked to the journalized datastore.
JRN_FULL_DATA_VIEW	Full name of the data view linked to the journalized datastore.
JRN_FULL_TRIGGER	Full name of the trigger linked to the journalized datastore.
JRN_FULL_ITRIGGER	Full name of the Insert trigger linked to the journalized datastore.
JRN_FULL_UTRIGGER	Full name of the Update trigger linked to the journalized datastore.
JRN_FULL_DTRIGGER	Full name of the Delete trigger linked to the journalized datastore.
JRN_SUBSCRIBER	Name of the subscriber table in the work schema .
JRN_NAME	Name of the journalized datastore.
JRN_VIEW	Name of the view linked to the journalized datastore.
JRN_DATA_VIEW	Name of the data view linked to the journalized datastore.
JRN_TRIGGER	Name of the trigger linked to the journalized datastore.
JRN_ITRIGGER	Name of the Insert trigger linked to the journalized datastore.
JRN_UTRIGGER	Name of the Update trigger linked to the journalized datastore.
JRN_DTRIGGER	Name of the Delete trigger linked to the journalized datastore.
JRN_SUBSCRIBER	Name of the subscriber.
JRN_COD_MODE	Code of the journalized data model.
JRN_METHOD	Journalizing Mode (consistent or simple).
CDC_SET_TABLE	Full name of the table containing list of CDC sets.
CDC_TABLE_TABLE	Full name of the table containing the list of tables journalized through CDC sets.
CDC_SUBS_TABLE	Full name of the table containing the list of subscribers to CDC sets.

CDC_OBJECTS_TABLE	Full name of the table containing the journalizing parameters and objects.
-------------------	--

Examples

The table being journalized is `<%=odiRef.getJrnInfo("FULL_TABLE_NAME")%>`

getModel() Method

Usage

```
public java.lang.String getModel(java.lang.String pPropertyName)
```

Description

This method returns information on the current data model during the processing of a personalized reverse engineering. The list of available data is described in the **pPropertyName values** table.

Note: This method may be used on the source connection (data server being reverse-engineered) as well as on the target connection (odi repository). On the target connection, only the properties independent from the context can be specified (for example, the schema and catalog names cannot be used)

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the possible values for pPropertyName:

Parameter value	Description
ID	Internal identifier of the current model
MOD_NAME	Name of the current model
LSHEMA_NAME	Name of the logical schema of the current model
MOD_TEXT	Description of the current model
REV_TYPE	Reverse engineering type: S for standard reverse, C for customize
REV_UPDATE	Update flag of the model
REV_INSERT	Insert flag for the model
REV_OBJ_PATT	Mask for the objects to reverse.

	List of object types to reverse-engineer for this model. This is a semicolon separated list of object types codes. Valid codes are:
REV_OBJ_TYPE	<ul style="list-style-type: none"> • T: Table • V: View • Q: Queue • SY: System table • AT: Table alias • SY: Synonym
TECH_INT_NAME	Internal name of the technology of the current model.
LAGENT_NAME	Name of the logical execution agent for the reverse engineering.
REV_CONTEXT	Execution context of the reverse
REV_ALIAS_LTRIM	Characters to be suppressed for the alias generation
CKM	Check Knowledge Module
RKM	Reverse-engineering Knowledge Module
SCHEMA_NAME	Physical Name of the data schema in the current reverse context
WSCHEMA_NAME	Physical Name of the work schema in the current reverse context
CATALOG_NAME	Physical Name of the data catalog in the current reverse context
WCATALOG_NAME	Physical Name of the work catalog in the current reverse context
<flexfield code>	Value of the flexfield for the current model.

Examples

Retrieve the list of tables that are part of the mask of objects to reverse:

```
select TABLE_NAME,
       RES_NAME,
       replace(TABLE_NAME, '<%=odiRef.getModel("REV_ALIAS_LTRIM")%>' , '')
       ALIAS,
       TABLE_DESC
from MY_TABLES
where
TABLE_NAME like '<%=odiRef.getModel("REV_OBJ_PATT")%>'
```

getNewColComment() Method

Usage

```
public java.lang.String getNewColComment()
```

Description

In an action, this method returns the new comment for the column being handled by the DDL command, in a *Modify column comment action*.

getNewTableComment() Method

Usage

```
public java.lang.String getNewTableComment ()
```

Description

In an action, this method returns the new comment for the table being handled by the DDL command, in a *Modify table comment action*.

getNotNullCol() Method

Usage

```
public java.lang.String getNotNullCol (java.lang.String pPropertyName)
```

Description

This method returns information relative to a not null column of a datastore during a check procedure. It is accessible from a Check Knowledge Module if the current task is tagged as "mandatory".

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	odi internal identifier for the current column.
COL_NAME	Name of the Not null column.
MESS	Standard error message.
<flexfield code>	Flexfield value for the current not null column.

Examples

```
insert into...
select *
from ...
where <%=odiRef.getNotNullCol("COL_NAME")%> is null
```

getObjectName() Method

Usage

```
public java.lang.String getObjectName (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLocation)

public java.lang.String getObjectName (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLogicalSchemaName,
    java.lang.String pLocation)

public java.lang.String getObjectName (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLogicalSchemaName,
    java.lang.String pContextName,
    java.lang.String pLocation)

public java.lang.String getObjectName (
    java.lang.String pObjectName,
    java.lang.String pLocation)

public java.lang.String getObjectName (
    java.lang.String pObjectName)
```

Description

Returns the complete name of a physical object, including its catalog and schema. The pMode parameter indicates the substitution mask to use.

The first syntax builds the object name according to the current logical schema in the current context.

The second syntax builds the name of the object according to the logical schema indicated in the pLogicalSchemaName parameter in the current context.

The third syntax builds the name from the logical schema and the context indicated in the pLogicalSchemaName and pContextName parameters.

The fourth syntax builds the object name according to the current logical schema in the current context, with the local object mask (pMode = "L").

The fifth syntax is equivalent to the fourth with pLocation = "D".

Parameters

Parameter	Type	Description
pMode	String	"L" use the local object mask to build the complete path of the object.

		"R" use the remote object mask to build the complete path of the object.
		Note: When using the remote object mask, getObjectMask always resolved the object name using the default physical schema of the remote server.
pObjectName	String	Every string that represents a valid resource name (table or file). This object name may be prefixed by a prefix code that will be replaced at run-time by the appropriate temporary object prefix defined for the physical schema.
pLogicalSchemaName	String	Name of the forced logical schema of the object.
pContextName	String	Forced context of the object
pLocation	String	"W" Returns the complete name of the object in the physical catalog and the "work" physical schema that corresponds to the specified tuple (context, logical schema).
		"D" Returns the complete name of the object in the physical catalog and the data physical schema that corresponds to the specified tuple (context, logical schema.)

Prefixes

It is possible to prefix the resource name specified in the pObjectName parameter by a prefix code to generate a odi temporary object name (Error or Integration table, journalizing trigger, etc.).

The list of prefixes are given below.

Prefix	Description
%INT_PRF	Prefix for integration tables (default value is "I\$").
%COL_PRF	Prefix for Loading tables (default value is "C\$").
%ERR_PRF	Prefix for error tables (default value is "E\$").
%JRN_PRF_TAB	Prefix for journalizing tables (default value is "J\$").
%INT_PRF_VIE	Prefix for journalizing view (default value is "JV\$").
%INT_PRF_TRG	Prefix for journalizing triggers (default value is "T\$").

Note that the temporary objects are usually created in the work physical schema. Therefore, pLocation should be set to "W" when using a prefix to create or access a temporary object.

Examples

If you have defined the physical schema:

Data catalog:	db_odi
Data schema:	dbo

Work catalog:	tempdb
Work schema:	temp_owner

and you have associated this physical schema to the logical schema: MSSQL_ODI in the context CTX_DEV

A call to	Returns
<code><%=odiRef.getObjectName("L", "EMP", "MSSQL_ODI", "CTX_DEV", "W") %></code>	tempdb.temp_owner.EMP
<code><%=odiRef.getObjectName("L", "EMP", "MSSQL_ODI", "CTX_DEV", "D") %></code>	db_odi.dbo.EMP
<code><%=odiRef.getObjectName("R", "%ERR_PRFEMP", "MSSQL_ODI", "CTX_DEV", "W") %></code>	MyServer.tempdb.temp_owner.E\$_EMP
<code><%=odiRef.getObjectName("R", "EMP", "MSSQL_ODI", "CTX_DEV", "D") %></code>	MyServer.db_odi.dbo.EMP

getObjectNameDefaultPSchema() Method

Usage

```
public java.lang.String getObjectNameDefaultPSchema (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLocation)

public java.lang.String getObjectNameDefaultPSchema (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLogicalSchemaName,
    java.lang.String pLocation)

public java.lang.String getObjectNameDefaultPSchema (
    java.lang.String pMode,
    java.lang.String pObjectName,
    java.lang.String pLogicalSchemaName,
    java.lang.String pContextName,
    java.lang.String pLocation)

public java.lang.String getObjectNameDefaultPSchema (
    java.lang.String pObjectName,
    java.lang.String pLocation)

public java.lang.String getObjectNameDefaultPSchema (
    java.lang.String pObjectName)
```

Description

The method is similar to the getObjectName method. However, the object name is computed for the default physical schema of the data server to which the physical schema is attached. In getObjectName, the object name is computed for the physical schema itself.

- For more information, see getObjectName.

getPK() Method

Usage

```
public java.lang.String getPK(java.lang.String pPropertyName)
```

Description

This method returns information relative to the primary key of a datastore during a check procedure.

In an action, this method returns information related to the primary key currently handled by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

The following table lists the different possible values for pPropertyName

Parameter value	Description
ID	Internal number of the PK constraint.
KEY_NAME	Name of the primary key
MESS	Error message relative to the primary key constraint.
FULL_NAME	Full name of the PK generated with the local object mask.
<flexfield code>	Flexfield value for the primary key.

Examples

The primary key of my table is called: `<%=odiRef.getPK("KEY_NAME")%>`

getPKColList() Method

Usage

```
public java.lang.String getPKColList( java.lang.String pStart,  
java.lang.String pPattern,  
java.lang.String pSeparator,  
java.lang.String pEnd)
```

Description

Offers a list of columns and expressions for the primary key being checked.

The pPattern parameter is interpreted and then repeated for each element of the list. It is separated from its predecessor by the pSeparator parameter. The generated string starts with pStart and ends with pEnd.

This list contains an element for each column of the current primary key. It is accessible from a Check Knowledge Module if the current task is tagged as an "primary key".

In an action, this method returns the list of the columns of the primary key handled by the DDL command, ordered by their position in the key.

Parameters

Parameters	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of attributes that can be used in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each attribute occurrence in the pattern sequence is replaced with its value. The attributes must be between brackets. ([and])</p> <p>Example « My string [COL_NAME] is a column »</p>
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as their associated description.

Parameter value	Description
I_COL	Column internal identifier
COL_NAME	Name of the key column
COL_HEADING	Header of the key column
COL_DESC	Column description
POS	Position of the column
LONGC	Length (Precision) of the column
SCALE	Scale of the column
FILE_POS	Beginning position of the column (fixed file)
BYTES	Number of physical bytes of the column
FILE_END_POS	End of the column (FILE_POS + BYTES)

IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: non null)
CHECK_FLOW	Flow control flag for of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol for the column
REC_CODE_LIST	List of the record codes retained for the column
COL_NULL_IF_ERR	Processing flag for the column (0 = Reject, 1 = Set active trace to null , 2= Set inactive trace to null)
DEF_VALUE	Default value for the column
EXPRESSION	Not used
CX_COL_NAME	Not used
ALIAS_SEP	Grouping symbol used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
<flexfield code>	Flexfield value for the current column.

Examples

If the CUSTOMER table has an primary key PK_CUSTOMER (CUST_ID, CUST_NAME) and you want to generate the following code:

```
create table T_PK_CUSTOMER (CUST_ID numeric(10) not null, CUST_NAME
varchar(50) not null)
```

you just have to write:

```
create table T_<%=odiRef.getPK("KEY_NAME")%> <%=odiRef.getPKColList("(" ,
"[COL_NAME] [DEST_CRE_DT] not null", ", ", ", ")")%>
```

Explanation: the getPKColList function will be used to generate the part (CUST_ID numeric(10) not null, CUST_NAME varchar(50) not null), which starts and stops with a parenthesis and repeats the pattern (column, a data type, and not null) separated by commas for each column of the primary key. Thus

- the first parameter "(" of the function indicates that we want to start the string with the string "("
- the second parameter "[COL_NAME] [DEST_CRE_DT] not null" indicates that we want to repeat this pattern for each column of the primary key. The keywords [COL_NAME] and [DEST_CRE_DT] reference valid keywords of the Pattern Attributes List table
- the third parameter ", " indicates that we want to separate interpreted occurrences of the pattern with the string ", "
- the forth parameter ")" of the function indicates that we want to end the string with the string ")"

getPop() Method

Usage

```
public java.lang.String getPop(java.lang.String pPropertyName)
```

Description

Generic method that returns general information of the current interface. The list of available information is described in the **pPropertyName values** table.

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the different possible values for pPropertyName:

Parameter value	Description
I_POP	Internal number of the interface.
FOLDER	Name of the folder of the interface
POP_NAME	Name of the interface
IND_WORK_TARG	Position flag of the staging area.
LSHEMA_NAME	Name of the logical schema which is the staging area of the interface
DESCRIPTION	Description of the interface
WSTAGE	Flag indicating the nature of the target datastore:

	<ul style="list-style-type: none">• E - target datastore is an existing table (not a temporary table).• N - target datastore is a temporary table in the data schema.• W - target datastore is a temporary table in the work schema.
TABLE_NAME	Target table name
KEY_NAME	Name of the update key
DISTINCT_ROWS	Flag for doubles suppression
OPT_CTX	Name of the optimization context of the interface
TARG_CTX	Name of the execution context of the interface
MAX_ERR	Maximum number of accepted errors
MAX_ERR_PRCT	Error indicator in percentage
IKM	Name of the Integration Knowledge Module
LKM	Name of the Loading Knowledge module
CKM	Name of the Check Knowledge Module
HAS_JRN	Returns 1 if there is a journalized table in source of the interface, 0 otherwise.
<flexfield code>	Flexfield value for the interface.

Examples

The current interface is: `<%=odiRef.getPop("POP_NAME")%>` and runs on the logical schema: `<%=odiRef.getInfo("L_SCHEMA_NAME")%>`

getPrevStepLog() Method

Usage

```
public java.lang.String getPrevStepLog(java.lang.String pPropertyName)
```

Description

Returns information about the most recently executed step in a package. The information requested is specified through the pPropertyName parameter. If there is no previous step (for example, if the getPrevStepLog step is executed from outside a package), the exception "No previous step" is raised.

Parameters

Parameter	Type	Description
-----------	------	-------------

pPropertyName String String that contains the name of the requested property about the previous step. See the list of valid properties below.

Values for pPropertyName

The following table lists the different possible values for pPropertyName:

Parameter value	Description
SESS_NO	The number of the session.
NNO	The number of the step within a package. The first step executed is 0.
STEP_NAME	The name of the step.
	A code indicating the type of step. The following values may be returned:
	<ul style="list-style-type: none"> • F: Interface • VD: Variable declaration • VS: Set/Increment variable • VE: Evaluate variable • V: Refresh variable • T: Procedure
STEP_TYPE	<ul style="list-style-type: none"> • OE: OS command • SE: odi Tool • RM: Reverse-engineer model • CM: Check model • CS: Check sub-model • CD: Check datastore • JM: Journalize model • JD: Journalize datastore
CONTEXT_NAME	The name of the context in which the step was executed.
MAX_ERR	The maximum number or percentage of errors tolerated.
MAX_ERR_PRCT	Returns 1 if the maximum number of errors is expressed as a percentage, 0 otherwise.
RUN_COUNT	The number of times this step has been executed.
BEGIN	The date and time that the step began.
END	The date and time that the step terminated.
DURATION	Time the step took to execute in seconds.
STATUS	<p>Returns the one-letter code indicating the status with which the previous step terminated. The state R (Running) is never returned.</p> <ul style="list-style-type: none"> • D: Done (success)

- E: Error
- Q: Queued
- W: Waiting
- M: Warning

RC	Return code. 0 indicates no error.
MESSAGE	Error message returned by previous step, if any. Blank string if no error.
INSERT_COUNT	Number of rows inserted by the step.
DELETE_COUNT	Number of rows deleted by the step.
UPDATE_COUNT	Number of rows updated by the step.
ERROR_COUNT	Number of erroneous rows detected by the step, for quality control steps.

Examples

Previous step '`<%=odiRef.getPrevStepLog("STEP_NAME")%>`' executed in '`<%=odiRef.getPrevStepLog("DURATION")%>`' seconds.

getQuotedString()

Usage

```
public java.lang.String getQuotedString(java.lang.String pString)
```

Description

This method returns a string surrounded with quotes. It preserves quotes and escape characters such as `\n`, `\t` that may appear in the string.

This method is useful to protect a string passed as a value in Java or Jython code.

Parameters

Parameter	Type	Description
pString	String	String that to be protected with quotes.

Examples

In the following Java code, the `getQuotedString` method is used to generate a valid string value.

```
String condSqlOK = <%=odiRef.getQuotedString(odiRef.getCK("MESS"))%>;  
String condSqlKO = <%=odiRef.getCK("MESS")%>;
```

If the message for the condition is "Error:\n Zero is not a valid value", the generated code is as shown below. Without the `getQuotedString`, the code is incorrect, as the \n is not preserved and becomes a carriage return.

```
String condSqlOK = "Error:\n Zero is not a valid value";
String condSqlKO = "Error:
Zero is not a valid value";
```

getOption() Method getUserExit() Method

Usage

```
public java.lang.String getOption(java.lang.String pOptionName)
public java.lang.String getUserExit(java.lang.String pOptionName)
```

Description

Returns the value of an Option (also known as a User Exit) of a KM or Procedure.
The `getUserExit` syntax is deprecated and is only kept for compatibility reasons.

Parameters

Parameter	Type	Description
pOptionName	String	String that contains the name of the requested option.

Examples

The value of my MY_OPTION_1 option is <%=odiRef.**getOption**("MY_OPTION_1")%>

getSchemaName() Method

Usage

```
public java.lang.String getSchemaName(
java.lang.String pLogicalSchemaName,
java.lang.String pLocation)
public java.lang.String getSchemaName(
java.lang.String pLogicalSchemaName,
java.lang.String pContextCode,
java.lang.String pLocation)
public java.lang.String getSchemaName(
java.lang.String pLocation)
public java.lang.String getSchemaName()
```

Description

Retrieves the physical name of a data schema or work schema from its logical schema.

If the first syntax is used, the returned schema corresponds to the current context.

If the second syntax is used, the returned schema corresponds to context specified in the pContextCode parameter.

The third syntax returns the name of the data schema (D) or work schema (W) for the current logical schema in the current context.

The fourth syntax returns the name of the data schema (D) for the current logical schema in the current context.

Parameters

Parameter	Type	Description
pLogicalSchemaName	String	Name of the logical schema of the schema
pContextCode	String	Forced context of the schema
pLocation	String	"W" Returns the work schema of the physical schema that corresponds to the tuple (context, logical schema)
		"D" Returns the data schema of the physical schema that corresponds to the tuple (context, logical schema)

Examples

If you have defined the physical schema:

Pluton.db_odi.dbo

Data catalog:	db_odi
Data schema:	dbo
Work catalog:	tempdb
Work schema:	temp_owner

and you have associated this physical schema to the logical schema: MSSQL_ODI in the context CTX_DEV

The call to	Returns
<code><%=odiRef.getSchemaName("MSSQL_ODI", "CTX_DEV", "W") %></code>	temp_owner
<code><%=odiRef.getSchemaName("MSSQL_ODI", "CTX_DEV", "D") %></code>	dbo

GetSchemaNameDefaultPSchema() Method

Usage


```

public java.lang.String getSchemaNameDefaultPSchema (
    java.lang.String pLogicalSchemaName,
    java.lang.String pLocation)

public java.lang.String getSchemaNameDefaultPSchema (
    java.lang.String pLogicalSchemaName,
    java.lang.String pContextCode,
    java.lang.String pLocation)

public java.lang.String getSchemaNameDefaultPSchema (
    java.lang.String pLocation)

public java.lang.String getSchemaNameDefaultPSchema ()

```

Description

Allows you to retrieve the name of the **default** physical data schema or work schema for the data server to which is associated the physical schema corresponding to the tuple (logical schema, context). If no context is specified, the current context is used. If no logical schema name is specified, then the current logical schema is used. If no pLocation is specified, then the data schema is returned.

Parameters

Parameter	Type	Description
pLogicalSchemaName	String	Name of the logical schema
pContextCode	String	Code of the enforced context of the schema
pLocation	String	"W" Returns the work schema of the default physical schema associate to the data server to which the physical schema corresponding to the tuple (context, logical schema) is also attached.
		"D" Returns the data schema of the physical schema corresponding to the tuple (context, logical schema)

Examples

If you have defined the physical schemas:

`Pluton.db_odi.dbo`

Data catalog:	db_odi
Data schema:	dbo
Work catalog:	tempdb
Work schema:	temp_odi
Default Schema	Yes

that you have associated with this physical schema: MSSQL_ODI in the context CTX_DEV, and `Pluton.db_doc.doc`

Data catalog:	db_doc
Data schema:	doc
Work catalog:	tempdb
Work schema:	temp_doc
Default Schema	No

that you have associated with this physical schema: MSSQL_DOC in the context CTX_DEV

The call to	Returns
<code><%=odiRef.getSchemaNameDefaultPSchema("MSSQL_DOC", "CTX_DEV", "W") %></code>	temp_odi
<code><%=odiRef.getSchemaNameDefaultPSchema("MSSQL_DOC", "CTX_DEV", "D") %></code>	dbo

getSession() Method

Usage

```
public java.lang.String getSession(java.lang.String pPropertyName)
```

Description

Generic method returning general information about the current session. The list of available properties is described in the **pPropertyName values** table.

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the different possible values for pPropertyName:

Parameter value	Description
SESS_NO	Internal number of the session
SESS_NAME	Name of the session
SCEN_VERSION	Current scenario version
CONTEXT_NAME	Name of the execution context

CONTEXT_CODE	Code of the execution context
AGENT_NAME	Name of the physical agent in charge of the execution
SESS_BEG	Date and time of the beginning of the session
USER_NAME	odi User running the session.

Examples

The current session is: `<%=odiRef.getSession("SESS_NAME")%>`

getSessionVarList() Method

Usage

```
public java.lang.String getSessionVarList( java.lang.String pStart,  
java.lang.String pPattern,  
java.lang.String pSeparator,  
java.lang.String pEnd,  
java.lang.String pSelector)
```

Description

Reserved for future use.

Parameters

Reserved for future use.

Examples

Reserved for future use.

getSrcColList() Method

Usage

```
public java.lang.String getSrcColList( java.lang.String pStart,  
java.lang.String pUnMappedPattern,  
java.lang.String pMappedPattern,  
java.lang.String pSeparator,  
java.lang.String pEnd)
```

Description

This method available in LKMs and IKMs, returns properties for a list of columns. This list includes all the columns of the sources processed by the LKM (from the source) or the IKM (from the staging area). The list is sorted on the column position in the source tables.

The properties displayed depend on whether the column is mapped or not. If the column is mapped, the properties returned are defined in the `pMappedPattern` pattern. If the column is not mapped, the properties returned are defined in the `pUnMappedPattern` pattern.

The attributes usable in a pattern are detailed in "Pattern Attributes List". Each occurrence of the attributes in the pattern string is replaced by its value. Attributes must be between brackets ([and]). Example: "My string [COL_NAME] is a column".

The `pMappedPattern` or `pUnMappedPattern` parameter is interpreted and then repeated for each element of the list. Patterns are separated with `pSeparator`. The generated string begins with `pStart` and ends with `pEnd`.

If there is a journalized datastore in the source of the interface, the three journalizing pseudo columns `JRN_FLG`, `JRN_DATE`, and `JRN_SUBSCRIBER` are added as columns of the journalized source datastore.

Parameters

Parameter	Type	Description
<code>pStart</code>	String	This sequence marks the beginning of the string to generate.
<code>pUnMappedPattern</code>	String	The pattern is repeated for each occurrence in the list if the column is not mapped.
<code>pMappedPattern</code>	String	The pattern is repeated for each occurrence in the list, if the column is mapped.
<code>pSeparator</code>	String	This parameter separates patterns.
<code>pEnd</code>	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists different parameters values as well as their associated description.

Parameter value	Description
<code>I_COL</code>	Internal identifier of the column
<code>COL_NAME</code>	Name of the column
<code>ALIAS_NAME</code>	Name of the column. Unlike <code>COL_NAME</code> , this attribute returns the column name without the optional technology delimiters. These delimiters appear when the column name contains for instance spaces.
<code>COL_HEADING</code>	Header of the column
<code>COL_DESC</code>	Description of the column
<code>POS</code>	Position of the column
<code>LONGC</code>	Column length (Precision)

SCALE	Scale of the column
FILE_POS	Beginning (index) of the column
BYTES	Number of physical bytes in the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: not null)
CHECK_FLOW	Flow control flag of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column
COL_DEC_SEP	Decimal symbol of the column
REC_CODE_LIST	List of the record codes retained in the column
COL_NULL_IF_ERR	Processing flag of the column (0 = Reject, 1 = Set to null active trace, 2= set to null inactive trace)
DEF_VALUE	Default value of the column
EXPRESSION	Text of the expression (as typed in the mapping field) executed on the source (LKM) or the staging area (IKM). If the column is not mapped, this parameter returns an empty string.
CX_COL_NAME	<i>Not supported.</i>
ALIAS_SEP	Separator used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target (IKM) or staging area (LKM) technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
MANDATORY_CLAUSE	Returns <code>NOT NULL</code> if the column is mandatory. Otherwise, returns the null keyword for the technology.
DEFAULT_CLAUSE	Returns <code>DEFAULT <default value></code> if any default value exists.

	Otherwise, returns an empty string.
COL_DESC	Description (comment) of the column. Quotes and double quotes are replaced with spaces.
<flexfield code>	Flexfield value for the current column.

Examples

To create a table similar to a source file:

```
create table <%=odiRef.getTable("L", "COLL_NAME", "D")%>_F
(
  <%=odiRef.getSrcColList(" ", "[COL_NAME] [DEST_CRE_DT] ", "[COL_NAME]
  [DEST_CRE_DT] ", "\n", "")%>
)
```

getSrcTablesList() Method

Usage

```
public java.lang.String getSrcTablesList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getSrcTablesList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Offers a list of source tables of an interface. This method can be used to build a FROM clause in a SELECT order. However, it is advised to use the getFrom() method instead.

The pPattern pattern is interpreted and then repeated for each element of the list and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This parameter marks the beginning of the string to generate.
		The pattern is repeated for each occurrence in the list.
pPattern	String	The list of possible attributes in a pattern is detailed in the table « Pattern Attributes List »
		Each attribute occurrence in the pattern string is substituted with its value.
		The attributes must be between brackets ([and])

Example « My string [COL_NAME] is a column »

pSeparator String This parameter separates each pattern from its predecessor.

pEnd String This parameter marks the end of the string to generate.

Pattern Attributes List

The following table lists the different values of the parameters as well as the associated description.

Attribute	Description
I_TABLE	odi internal number of the current source table if available.
MODEL_NAME	Name of the model of the current source table, if available.
SUB_MODEL_NAME	Name of the sub-model of the current source table, if available
TECHNO_NAME	Name of the technology of the source datastore
LSHEMA_NAME	Logical schema of the source table
TABLE_NAME	Logical name of the source datastore
RES_NAME	Physical access name of the resource (file name or JMS queue, physical name of the table, etc.). If there is a journalized datastore in source of the interface, the source table is the clause is replaced by the data view linked to the journalized source datastore.
CATALOG	Catalog of the source datastore (resolved at runtime)
WORK_CATALOG	Work catalog of the source datastore
SCHEMA	Schema of the source datastore (resolved at runtime)
WORK_SCHEMA	Work schema of the source datastore
TABLE_ALIAS	Alias of the datastore as it appears in the tables list, if available
POP_TAB_ALIAS	Alias of the datastore as it appears in the current interface, if available.
TABLE_TYPE	Type of the datastore source, if available.
DESCRIPTION	Description of the source datastore, if available.
R_COUNT	Number of records of the source datastore,if available.
FILE_FORMAT	File format, if available.
FILE_SEP_FIELD	Field separator (file)
XFILE_SEP_FIELD	Hexadecimal field separator (file)
SFILE_SEP_FIELD	Field separator string (file)

FILE_ENC_FIELD	Field beginning and ending character (file)
FILE_SEP_ROW	Record separator (file)
XFILE_SEP_ROW	Hexadecimal record separator (file)
SFILE_SEP_ROW	Record separator string (file)
FILE_FIRST_ROW	Number of header lines to ignore, if available.
FILE_DEC_SEP	Default decimal separator for the datastore, if available.
METADATA	Description in odi format of the metadata of the current resource, if available.
OLAP_TYPE	OLAP type specified in the datastore definition
IND_JRN	Flag indicating that the datastore is including in CDC.
JRN_ORDER	Order of the datastore in the CDC set for consistent journalizing.
<flexfield code>	Flexfield value for the current table.

Examples

```
insert into <%=odiRef.getTable("L", "COLL_NAME", "W")%>
select <%=odiRef.getColList("", "[EXPRESSION]", "", "", "INS=1")%>
from <%=odiRef.getSrcTablesList("", "[CATALOG].[SCHEMA].[TABLE_NAME] as
[POP_TAB_ALIAS]", "", "", "")%>
where (1=1)
<%=odiRef.getJoinList("and ", "([EXPRESSION])", " and ", "")%>
<%=odiRef.getFilterList("and ", "([EXPRESSION])", " and ", "")%>
<%=odiRef.getGrpBy()%>
<%=odiRef.getHaving()%>
```

Explanation: the `getSrcTablesList` function will be used to generate the FROM clause of the SELECT STATEMENT that repeats the pattern (CATALOG.SCHEMA.TABLE_NAME as POP_TAB_ALIAS) separated by commas for each table in source.

- The first parameter "" of the function indicates that we want do not want to start the string with any specific character.
- The second parameter "[CATALOG].[SCHEMA].[TABLE_NAME] as [POP_TAB_ALIAS]" indicates that we want to repeat this pattern for each source table. The keywords [CATALOG], [SCHEMA], [TABLE_NAME] and [POP_TAB_ALIAS] reference valid keywords of the table Pattern Attributes List
- The third parameter", " indicates that we want to separate each interpreted occurrence of the pattern with the string ", "
- The fourth parameter "" of the function indicates that we want to end the string with no specific character

getStep() Method

Usage


```
public java.lang.String getStep(java.lang.String pPropertyName)
```

Description

Generic method that returns general information on the current step. The list of available information is described in the **pPropertyName values** table

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the possible values for pPropertyName:

Parameter value	Description
SESS_NO	Number of the session to which the step belongs.
NNO	Number of the step in the session
NB_RUN	Number of execution attempts
STEP_NAME	Step name
STEP_TYPE	Step type
CONTEXT_NAME	Name of the execution context
VAR_INCR	Step variable increment
VAR_OP	Operator used to compare the variable
VAR_VALUE	Forced value of the variable
OK_EXIT_CODE	Exit code in case of success
OK_EXIT	End the package in case of success
OK_NEXT_STEP	Next step in case of success.
OK_NEXT_STEP_NAME	Name of the next step in case of success
KO_RETRY	Number of retry attempts in case of failure.
KO_RETRY_INTERV	Interval between each attempt in case of failure
KO_EXIT_CODE	Exit code in case of failure.
KO_EXIT	End the package in case of failure.
KO_NEXT_STEP	Next step in case of failure.
KO_NEXT_STEP_NAME	Name of the next step in case of failure

Examples

The current step is: <%=odiRef.getStep("STEP_NAME")%>

getSubscriberList() Method

Usage

```
public java.lang.String getSubscriberList( java.lang.String pStart,  
java.lang.String pPattern,  
java.lang.String pSeparator,  
java.lang.String pEnd)
```

Alternative syntax:

```
public java.lang.String getSubscriberList(  
java.lang.String pPattern,  
java.lang.String pSeparator)
```

Description

Offers a list of subscribers for a journalized table. The pPattern parameter is interpreted and then repeated for each element of the list, and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

In the alternative syntax, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	<p>The pattern is repeated for each occurrence in the list.</p> <p>The list of the attributes usable in a pattern is detailed in the table « Pattern Attributes List »</p> <p>Each occurrence of the attributes in the pattern string is replaced by its value. Attributes must be between brackets ([and])</p> <p>Example « My name is [SUBSCRIBER]»</p>
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.

Pattern Attributes List

The following table lists different parameters values as well as their associated description.

Parameter value	Description
SUBSCRIBER	Name of the Subscriber

Examples

Here is list of Subscribers: <%=odiRef.getSubscriberList("\nBegin List\n", "- [SUBSCRIBER]", "\n", "\nEnd of List\n")%>

getSysDate() Method

Usage

```
public java.lang.String getSysDate()  
public java.lang.String getSysDate(pDateFormat)
```

Description

This method returns the system date of the machine running the session.

Parameters

Parameter	Type	Description
pDateFormat	String	Date format used to return the system date. For more information on the date format patterns, see Date Format in odi.

Examples

Current year is: :<%=odiRef.getSysDate("y")%>

getTable() Method

Usage

```
public java.lang.String getTable(  
    java.lang.String pMode,  
    java.lang.String pProperty,  
    java.lang.String pLocation)  
public java.lang.String getTable(  
    java.lang.String pProperty,  
    java.lang.String pLocation)  
public java.lang.String getTable(  
    java.lang.String pProperty)
```

Description

Allows the retrieval of the full name of temporary and permanent tables handled by odi.

Parameters

Parameter	Type	Description																																		
pMode	String	"L" uses the local object mask to build the complete path of the object. This value is used when pMode is not specified.																																		
		"R" uses the object mask to build the complete path of the object																																		
		"A" Automatic. Defines automatically the adequate mask to use.																																		
		Parameter that indicates the name of the table to be built. The list of possible values is provided hereafter:																																		
pProperty	String	<table><tr><th>Parameter value</th><th>Description</th></tr><tr><td>ID</td><td>Datastore identifier.</td></tr><tr><td>TARG_NAME</td><td>Full name of the target datastore. In actions, this parameter returns the name of the current table handled by the DDL command</td></tr><tr><td>COLL_NAME</td><td>Full name of the loading datastore.</td></tr><tr><td>INT_NAME</td><td>Full name of the integration datastore.</td></tr><tr><td>ERR_NAME</td><td>Full name of the error datastore.</td></tr><tr><td>CHECK_NAME</td><td>Name of the error summary datastore.</td></tr><tr><td>CT_NAME</td><td>Full name of the checked datastore.</td></tr><tr><td>FK_PK_TABLE_NAME</td><td>Full name of the datastore referenced by a Foreign Key</td></tr><tr><td>JRN_NAME</td><td>Full name of the journalized datastore.</td></tr><tr><td>JRN_VIEW</td><td>Full name of the view linked to the journalized datastore.</td></tr><tr><td>JRN_DATA_VIEW</td><td>Full name of the data view linked to the journalized datastore.</td></tr><tr><td>JRN_TRIGGER</td><td>Full name of the trigger linked to the journalized datastore.</td></tr><tr><td>JRN_ITRIGGER</td><td>Full name of the Insert trigger linked to the journalized datastore.</td></tr><tr><td>JRN_UTRIGGER</td><td>Full name of the Update trigger linked to the journalized datastore.</td></tr><tr><td>JRN_DTRIGGER</td><td>Full name of the Delete trigger linked to the journalized datastore.</td></tr><tr><td>SUBSCRIBER_TABLE</td><td>Full name of the datastore containing the subscribers list.</td></tr></table>	Parameter value	Description	ID	Datastore identifier.	TARG_NAME	Full name of the target datastore. In actions, this parameter returns the name of the current table handled by the DDL command	COLL_NAME	Full name of the loading datastore.	INT_NAME	Full name of the integration datastore.	ERR_NAME	Full name of the error datastore.	CHECK_NAME	Name of the error summary datastore.	CT_NAME	Full name of the checked datastore.	FK_PK_TABLE_NAME	Full name of the datastore referenced by a Foreign Key	JRN_NAME	Full name of the journalized datastore.	JRN_VIEW	Full name of the view linked to the journalized datastore.	JRN_DATA_VIEW	Full name of the data view linked to the journalized datastore.	JRN_TRIGGER	Full name of the trigger linked to the journalized datastore.	JRN_ITRIGGER	Full name of the Insert trigger linked to the journalized datastore.	JRN_UTRIGGER	Full name of the Update trigger linked to the journalized datastore.	JRN_DTRIGGER	Full name of the Delete trigger linked to the journalized datastore.	SUBSCRIBER_TABLE	Full name of the datastore containing the subscribers list.
		Parameter value	Description																																	
		ID	Datastore identifier.																																	
		TARG_NAME	Full name of the target datastore. In actions, this parameter returns the name of the current table handled by the DDL command																																	
		COLL_NAME	Full name of the loading datastore.																																	
		INT_NAME	Full name of the integration datastore.																																	
		ERR_NAME	Full name of the error datastore.																																	
		CHECK_NAME	Name of the error summary datastore.																																	
		CT_NAME	Full name of the checked datastore.																																	
		FK_PK_TABLE_NAME	Full name of the datastore referenced by a Foreign Key																																	
		JRN_NAME	Full name of the journalized datastore.																																	
		JRN_VIEW	Full name of the view linked to the journalized datastore.																																	
		JRN_DATA_VIEW	Full name of the data view linked to the journalized datastore.																																	
		JRN_TRIGGER	Full name of the trigger linked to the journalized datastore.																																	
		JRN_ITRIGGER	Full name of the Insert trigger linked to the journalized datastore.																																	
		JRN_UTRIGGER	Full name of the Update trigger linked to the journalized datastore.																																	
JRN_DTRIGGER	Full name of the Delete trigger linked to the journalized datastore.																																			
SUBSCRIBER_TABLE	Full name of the datastore containing the subscribers list.																																			

pLocation String	CDC_SET_TABLE		Full name of the table containing list of CDC sets.
	CDC_TABLE_TABLE		Full name of the table containing the list of tables journalized through CDC sets.
	CDC_SUBS_TABLE		Full name of the table containing the list of subscribers to CDC sets.
	CDC_OBJECTS_TABLE		Full name of the table containing the journalizing parameters and objects.
	<flexfield code>		Flexfield value for the current target table.
	"W"	Returns the full name of the object in the physical catalog and the physical work schema that corresponds to the current tuple (context, logical schema)	
	"D"	Returns the full name of the object in the physical catalog and the physical data schema that corresponds to the current tuple (context, logical schema).	
	"A"	Lets odi determine the default location of the object. This value is used if pLocation is not specified.	

Examples

If you have defined the following elements:

physical schema: Pluton.db_odi.dbo

Data catalog:	db_odi
Data schema:	dbo
Work catalog:	tempdb
Work schema:	temp_owner
Local Mask:	%CATALOG.%SCHEMA.%OBJECT
Remote mask:	%DSERVER:%CATALOG.%SCHEMA.%OBJECT
Loading prefix:	CZ_
Error prefix:	ERR_
Integration prefix:	I\$_

and you have associated this physical schema to the logical schema: MSSQL_ODI in your context CTX_DEV

and you table is named CUSTOMER

A call to	Returns
<code><%=odiRef.getTable("L", "COLL_NAME",</code>	<code>tempdb.temp_owner.CZ_0CUSTOMER</code>

```
"W") %>

<%=odiRef.getTable("R", "COLL_NAME",          MyServer:db_odi.dbo.CZ_0CUSTOMER
"D") %>

<%=odiRef.getTable("L", "INT_NAME",           tempdb.temp_owner.I$_CUSTOMER
"W") %>

<%=odiRef.getTable("R", "ERR_NAME",          MyServer:db_odi.dbo.ERR_CUSTOMER
"D") %>
```

getTargetColList() Method

Usage

```
public java.lang.String getTargetColList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd,
java.lang.String pSelector)
```

Alternative syntaxes:

```
public java.lang.String getTargetColList( java.lang.String pStart,
java.lang.String pPattern,
java.lang.String pSeparator,
java.lang.String pEnd)
```

```
public java.lang.String getTargetColList(
java.lang.String pPattern,
java.lang.String pSeparator)
```

Description

Provides a list of columns for the interface's target table.

The pPattern parameter is interpreted and then repeated for each element of the list (selected according to pSelector parameter) and separated from its predecessor with the parameter pSeparator. The generated string begins with pStart and ends with pEnd.

In the alternative syntaxes, any parameters not set are set to an empty string.

Parameters

Parameter	Type	Description
pStart	String	This sequence marks the beginning of the string to generate.
pPattern	String	The pattern is repeated for each occurrence in the list.
		The list of the attributes usable in a pattern is detailed in the table « Pattern Attributes List »
		Each occurrence of the attributes in the pattern string is replaced by its value. Attributes must be between brackets ([and])

Example « My string [COL_NAME] is a column of the target »		
pSeparator	String	This parameter separates each pattern from its predecessor.
pEnd	String	This sequence marks the end of the string to generate.
String that designates a Boolean expression that allows to filter the elements of the initial list with the following format :		
<SELECTOR> <Operator> <SELECTOR> etc. Parenthesis are authorized.		
Authorized operators:		
pSelector	String	1. No: NOT or !
		2. Or: OR or
		3. And: AND or &&
Example: (INS AND UPD) OR TRG		
The description of valid selectors is provided in the table « Selectors Description »		

Pattern Attributes List

The following table lists different parameters values as well as their associated description.

Parameter value	Description
I_COL	Internal identifier of the column
COL_NAME	Name of the column
COL_HEADING	Header of the column
COL_DESC	Description of the column
POS	Position of the column
LONGC	Column length (Precision)
SCALE	Scale of the column
FILE_POS	Beginning (index) of the column
BYTES	Number of physical bytes in the column
FILE_END_POS	End of the column (FILE_POS + BYTES)
IND_WRITE	Write right flag of the column
COL_MANDATORY	Mandatory character of the column (0: null authorized, 1: not null)
CHECK_FLOW	Flow control flag of the column (0: do not check, 1: check)
CHECK_STAT	Static control flag of the column (0: do not check, 1: check)
COL_FORMAT	Logical format of the column

COL_DEC_SEP	Decimal symbol of the column
REC_CODE_LIST	List of the record codes retained in the column
COL_NULL_IF_ERR	Processing flag of the column (0 = Reject, 1 = Set to null active trace, 2= set to null inactive trace)
DEF_VALUE	Default value of the column
ALIAS_SEP	Separator used for the alias (from the technology)
SOURCE_DT	Code of the column's datatype.
SOURCE_CRE_DT	Create table syntax for the column's datatype.
SOURCE_WRI_DT	Create table syntax for the column's writable datatype.
DEST_DT	Code of the column's datatype converted to a datatype on the target technology.
DEST_CRE_DT	Create table syntax for the column's datatype converted to a datatype on the target technology.
DEST_WRI_DT	Create table syntax for the column's writable datatype converted to a datatype on the target technology.
SCD_COL_TYPE	Behavior defined for the Slowly Changing Dimensions for this column in the data model.
MANDATORY_CLAUSE	Returns <code>NOT NULL</code> if the column is mandatory. Otherwise, returns the null keyword for the technology.
DEFAULT_CLAUSE	Returns <code>DEFAULT <default value></code> if any default value exists. Otherwise, returns an empty string.
COL_DESC	Description (comment) of the column. Quotes and double quotes are replaced with spaces.
JDBC_TYPE	Data Services - JDBC Type of the column returned by the driver.
<flexfield code>	Flexfield value for the current column.

Selectors Description

Parameters value	Description
INS	<ul style="list-style-type: none"> LKM: Not applicable IKM: Only for mapping expressions marked with insertion CKM: Not applicable
UPD	<ul style="list-style-type: none"> LKM: Not applicable IKM: Only for the mapping expressions marked with update

	<ul style="list-style-type: none"> • CKM: Non applicable
TRG	<ul style="list-style-type: none"> • LKM: Not applicable • IKM: Only for the mapping expressions executed on the target • CKM: Mapping expressions executed on the target.
NULL	<ul style="list-style-type: none"> • LKM: Not applicable • IKM: All mapping expressions loading not nullable columns • CKM: All target columns that do not accept null values
PK	<ul style="list-style-type: none"> • LKM: Not applicable • IKM: All mapping expressions loading the primary key columns • CKM: All the target columns that are part of the primary key
UK	<ul style="list-style-type: none"> • LKM: Not applicable. • IKM: All the mapping expressions loading the update key column chosen for the current interface. • CKM: Not applicable.
REW	<ul style="list-style-type: none"> • LKM: Not applicable. • IKM: All the mapping expressions loading the columns with read only flag not selected. • CKM: All the target columns with read only flag not selected.
MAP	<ul style="list-style-type: none"> • LKM: Not applicable • IKM: Not applicable • CKM: <ul style="list-style-type: none"> Flow control: All columns of the target table loaded with expressions in the current interface Static control: All columns of the target table
SCD_SK	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Surrogate Key in the data model definition.
SCD_NK	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Natural Key in the data model definition.
SCD_UPD	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Overwrite on Change in the data model definition.
SCD_INS	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Add Row on Change in the data model definition.
SCD_FLAG	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Current Record Flag in the data model definition.
SCD_START	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Starting Timestamp in the data model definition.
SCD_END	<ul style="list-style-type: none"> • LKM, CKM, IKM: All columns marked SCD Behavior: Ending Timestamp in

the data model definition.

WS_INS	• SKM : The column is flagged as allowing INSERT using Data Services.
WS_UPD	• SKM : The column is flagged as allowing UPDATE using Data Services.
WS_SEL	• SKM : The column is flagged as allowing SELECT using Data Services.

Examples

```
create table TARGET_COPY <%=odiRef.getTargetColList("(", "[COL_NAME]
[DEST_DT] null", " ", " ", ")", "")%>
```

getTargetTable() Method

Usage

```
public java.lang.String getTargetTable(java.lang.String pPropertyName)
```

Description

Generic method that returns general information on the current target table. The list of available data is described in the **pPropertyName values** table.

In an action, this method returns information on the table being processed by the DDL command.

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the possible values for pPropertyName:

Parameter value	Description
I_TABLE	Internal identifier of the datastore
MODEL_NAME	Name of the model of the current datastore.
SUB_MODEL_NAME	Name of the sub-model of the current datastore.
TECHNO_NAME	Name of the target technology.
LSHEMA_NAME	Name of the target logical schema.
TABLE_NAME	Name of the target datastore.
RES_NAME	Physical name of the target resource.

CATALOG	Catalog name.
WORK_CATALOG	Name of the work catalog.
SCHEMA	Schema name
WORK_SCHEMA	Name of the work schema.
TABLE_ALIAS	Alias of the current datastore.
TABLE_TYPE	Type of the datastore.
DESCRIPTION	Description of the current interface.
TABLE_DESC	Description of the current interface's target datastore. For a DDL command, description of the current table.
R_COUNT	Number of lines of the current datastore.
FILE_FORMAT	Format of the current datastore (file)
FILE_SEP_FIELD	Field separator (file)
XFILE_SEP_FIELD	Hexadecimal field separator (file)
SFILE_SEP_FIELD	Field separator string (file)
FILE_ENC_FIELD	Field beginning and ending character (file)
FILE_SEP_ROW	Record separator (file)
XFILE_SEP_ROW	Hexadecimal record separator (file)
SFILE_SEP_ROW	Record separator string (file)
FILE_FIRST_ROW	Number of lines to ignore at the beginning of the file (file)
FILE_DEC_SEP	Decimal symbol (file)
METADATA_DESC	Description of the metadata of the datastore (file)
OLAP_TYPE	OLAP type specified in the datastore definition
IND_JRN	Flag indicating that the datastore is including in CDC.
JRN_ORDER	Order of the datastore in the CDC set for consistent journalizing.
TABLE_DESC	Description (comment) of the table. The quotes and double quotes are replaced by spaces.
WS_NAME	Data Services - Name of the Web service generated for this datastore's model.
WS_NAMESPACE	Data Services - XML namespace of the web Service.
WS_JAVA_PACKAGE	Data Services - Java package generated for the web Service.

WS_ENTITY_NAME	Data Services - Entity name used for this datastore in the web service.
WS_DATA_SOURCE	Data Services - Datasource specified for this datastore's web service.
<flexfield code>	Flexfield value for the current table.

Examples

The current table is: `<%=odiRef.getTargetTable("RES_NAME")%>`

getUser() Method

Usage

```
public java.lang.String getUser(java.lang.String pPropertyName)
```

Description

Generic method returning general information about the user executing the current session. The list of available properties is described in the **pPropertyName values** table.

Parameters

Parameter	Type	Description
pPropertyName	String	String that contains the name of the requested property.

pPropertyName values

The following table lists the different possible values for pPropertyName:

Parameter Value	Description
I_USER	User identifier
USER_NAME	User name
IS_SUPERVISOR	Boolean flag indicating if the user is supervisor (1) or not (0).

Examples

This execution is performed by `<%=odiRef.getUser("USER_NAME")%>`

getOption() Method getUserExit() Method

Usage

```
public java.lang.String getOption(java.lang.String pOptionName)
```

```
public java.lang.String getUserExit(java.lang.String pOptionName)
```

Description

Returns the value of an Option (also known as a User Exit) of a KM or Procedure.
The getUserExit syntax is deprecated and is only kept for compatibility reasons.

Parameters

Parameter	Type	Description
pOptionName	String	String that contains the name of the requested option.

Examples

The value of my MY_OPTION_1 option is `<%=odiRef.getOption("MY_OPTION_1")%>`

hasPK() Method

Usage

```
public java.lang.Boolean hasPK()
```

Description

This method returns a boolean. The returned value is true if the datastore for which a web service is being generated has a primary key.

This method can only be used in SKMs.

Examples

```
<% if (odiRef.hasPK()) { %>
    There is a PK :
    <%=odiRef.getPK("KEY_NAME")%> : <%=odiRef.getPKColList("{",
    "\u0022[COL_NAME]\u0022", " ", " ", "}")%>
<% } else {%>
    There is NO PK.
<% } %>
```

isColAttrChanged() Method

Usage

```
public java.lang.Boolean
isColAttrChanged(java.lang.String pPropertyName)
```

Description

This method is usable in a column action for altering a column attribute or comment. It returns a boolean indicating if the column attribute passed as a parameter has changed.

Parameters

Parameter	Type	Description
pPropertyName	String	Attribute code (see below).

The following table lists the different possible values for pPropertyName

Parameter value	Description
DATATYPE	Column datatype, length or precision change,
LENGTH	Column length change (for example, VARCHAR(10) changes to VARCHAR(12)).
PRECISION	Column precision change (for example, DECIMAL(10,3) changes to DECIMAL(10,4)).
COMMENT	Column comment change.
NULL_TO_NOTNULL	Column nullable attribute change from NULL to NOT NULL.
NOTNULL_TO_NULL	Column nullable attribute change from NOT NULL to NULL.
NULL	Column nullable attribute change.
DEFAULT	Column default value change.

Examples

```
<% if (odiRef.IsColAttrChanged("DEFAULT") ) { %>
    /* Column default attribute has changed. */
<% } %>
```

nextAK() Method

Usage

```
public java.lang.Boolean nextAK()
```

Description

This method moves to the next alternate key (AK) of the datastore for which a Web service is being generated.

When first called, this method returns true and positions the current AK to the first AK of the datastore. If there is no AK for the datastore, it returns false.

Subsequent calls position the current AK to the next AKs of the datastore, and return true. If there is no next AK, the method returns false.

This method can be used only in SKMs.

Examples

In the example below, we iterate of all the AKs of the datastore. In each iteration of the while loop, the `getAK` and `getAKColList` methods return information on the various AKs of the datastore.

```
<% while (odiRef.nextAK()) { %>
    <%=odiRef.getAK("KEY_NAME")%>

    Columns <%=odiRef.getAKColList("{", "\u0022[COL_NAME]\u0022", " ",
    ", "}")%>
    Message : <%=odiRef.getAK("MESS")%>
<% } %>
```

nextCond() Method

Usage

```
public java.lang.Boolean nextCond()
```

Description

This method moves to the next condition (check constraint) of the datastore for which a Web service is being generated.

When first called, this method returns true and positions the current condition to the first condition of the datastore. If there is no condition for the datastore, it returns false.

Subsequent calls position the current condition to the next conditions of the datastore, and return true. If there is no next condition, the method returns false.

This method can be used only in SKMs.

Examples

In the example below, we iterate of all the conditions of the datastore. In each iteration of the while loop, the `getCK` method return information on the various conditions of the datastore.

```
<% while (odiRef.nextCond()) { %>
    <%=odiRef.getCK("COND_NAME")%>
    SQL :<%=odiRef.getCK("COND_SQL")%>
    MESS :<%=odiRef.getCK("MESS")%>
<% } %>
```

nextFK() Method

Usage

```
public java.lang.Boolean nextFK()
```

Description

This method moves to the next foreign key (FK) of the datastore for which a Web service is being generated.

When first called, this method returns true and positions the current FK to the first FK of the datastore. If there is no FK for the datastore, it returns false.

Subsequent calls position the current FK to the next FKs of the datastore, and return true. If there is no next FK, the method returns false.

This method can be used only in SKMs.

Examples

In the example below, we iterate of all the FKs of the datastore. In each iteration of the while loop, the `getFK` and `getFKColList` methods return information on the various FKs of the datastore.

```
<% while (odiRef.nextFK()) { %>
    FK : <%=odiRef.getFK("FK_NAME")%>
    Referenced Table : <%=odiRef.getFK("PK_TABLE_NAME")%>
    Columns <%=odiRef.getFKColList("{", "\u0022[COL_NAME]\u0022", ",
    ", "}")%>
    Message : <%=odiRef.getFK("MESS")%>
<% } %>
```

setNbInsert, setNbUpdate, setNbDelete, setNbErrors and setNbRows Methods

Usage

```
public java.lang.Void setNbInsert(public java.lang.Long)
public java.lang.Void setNbUpdate(public java.lang.Long)
public java.lang.Void setNbDelete(public java.lang.Long)
public java.lang.Void setNbErrors(public java.lang.Long)
public java.lang.Void setNbRows(public java.lang.Long)
```

Description

These methods set for the current task report the values for:

- the number of rows inserted (**setNbInsert**)
- the number of rows updated (**setNbUpdate**)
- the number of rows deleted (**setNbDelete**)
- the number of rows in error (**setNbErrors**)
- total number of rows handled during this task (**setNbRows**)

These numbers can be set independently from the real number of lines processed.

Important: This method can be used only within scripting engine commands, such as in Jython code and should not be enclosed in `<% %>` tags.

Examples

In the Jython example below, we set the number of inserted rows to the constant value of 50, and the number of erroneous rows to a value coming from an ODI variable called `#DEMO.NbErrors`.

```
InsertNumber=50
```

```
odiRef.setNbInsert (InsertNumber)
```

```
ErrorNumber=#DEMO.NbErrors
```

```
odiRef.setNbErrors (ErrorNumber)
```