



ART

Administrator Manual

Document version: 1.7b
Update Date: Sep, 16 2006
Web Site: <http://art.sourceforge.net>



Table of Contents

1. Introduction.....	3
1.1. Overview.....	3
1.2. Administration Overview.....	3
2. Data Sources.....	5
3. Groups.....	7
4. Users.....	8
4.1. Setup an User.....	8
4.2. Setup the Public User.....	9
5. Queries.....	10
5.1. Introduction.....	10
5.2. Header.....	11
5.3. Parameters.....	15
5.4. Execute a query via URL.....	18
5.5. Rules.....	18
5.6. An Example.....	18
5.7. Other Useful Queries.....	20
5.8. Limitations&Hints.....	21
6. Dynamic Queries.....	22
7. Portlets Containers.....	24
8. Text Objects.....	26
9. Scheduling.....	27
10. User Privileges.....	28
11. Admin Privileges.....	28
12. Rules.....	29
13. Chained Parameters.....	30



1. Introduction

1.1. Overview

ART is a J2EE web application created to quickly deploy SQL queries via web.

- *ART Administrators* define datasources, queries, parameters, groups, users, privileges, rules etc.
- *ART Users* can execute or schedule queries (specifying parameters), view the result set in a browser, download it in Excel xls, tsv, PDF, xml or view it as a graph (3d Bar, Pie, Line etc). Multiple objects can be grouped in a single page (Portlets Container).

ART is an open source software distributed under the GPL license. You can freely install and use it without any charge both for personal or commercial purposes.

The main ART objects:

Queries: standard SQL queries with dynamic parameters.

Portlets Containers: used to group queries in a single portal-like page: every query can be refreshed independently.

Text: small text fragments edited using an embedded WYSIWYG editor.

In portals terminology, a portlet is an independent frame that can be embedded in a web page.

1.2. Administration Overview

The Administrators define:

1. **Datasources** where to execute queries
2. **Groups** where to group queries
3. **Users**
4. **Objects** (Queries/Graphs, Text Objects, Portlets Containers)
5. **User Privileges** (who can execute a query)
6. **Admin Privileges**
7. **Rules** on queries (optional)

All the information is stored in set of tables - the *ART Repository*.

The normal flow when defining a new query object is:

1. Obtain the plain SQL statement and identify the parameters
2. If the target datasource is not already defined, define it
3. If new queries are needed in order to retrieve list of values (LOV) - to dynamically build parameter values list - define them
4. Define the query and the query parameters
5. Modify users privileges in order to allow users to execute the query

Once ART is configured (see quick install procedure on art web site), point your browser to `http://server_name:port/art` and follow the admin link. Log in




using your username/password (or the ART Repository username and password) and the ART Admin Console will appear. At least one group and one datasource must be defined prior to be able to create queries.

To configure external authentication sources (database, Windows Domain etc) refer to the ART Install Manual pdf document.

ART Admin part is multi-user: several administrators can concurrently modify the ART Repository on the fly - without interfere with users who are executing queries.

Admin Console:

Admin Console :: User Start Page :: Log Off Username [artu] :: Logged in at [1:07 PM]	
 ART Admin Console	
Properties	Define or update ART properties <i>(ART Repository, export path etc)</i>
Datasources	Define Datasources connection parameters <i>reinit and review connections status</i>
Groups	Define Groups names where to store objects
Rules	Define Rules names
Objects	Manage Queries, Graphs, Portlets, Texts <i>create, update, delete, copy SQL/Text statemens and portlets containers</i>
Users	Define Users
Users Privileges	Grant/Revoke to users the privilege to access an object
Admins Privileges	Grant/Revoke to junior/mid Admins the privilege to act on groups and datasources
Users Rules	Set users rule values



2.Data Sources

ART can execute queries any datasource (target databases) where a JDBC driver is available (see the Administrator Manual for further info, in Tomcat drivers can be placed in the common/lib directory). From the ART Admin Console click on the Datasource button. You reach the Manage Databases page. Choose to add a new database.

Manage Database	
Database connection parameters	
Database Name	ERP Production
Driver	oracle.jdbc.driver.OracleDrive
Database Url	jdbc:oracle:thin:@erpprod.com
Database Username	readonly
Database Password	*****
Submit	

You need to specify the following values:

- Target database name - it is a label that appear when executing queries
- JDBC driver name
- JDBC target database URL
- Username/Password of a database user in the target database (both values cannot be null). This user should have the correct grants to select from the tables you plan to use in the queries.

Click on the submit button. A connection test is performed and credentials are inserted in the ART Repository.

At the bottom of the Manage Databases page is a link to refresh and analyze the status of the connections performed by users - read the inline comments to learn more.

You may define several target databases (the ART Repository database itself can be a target database) with different JDBC drivers.



Some JDBC Drivers and URL:

Database: Oracle
Driver Name: oracle.jdbc.driver.OracleDriver
JDBC Url: jdbc:oracle:thin:@[serverName]:[port]:[sid]
Driver Available from: <http://otn.oracle.com>

Database: MySQL
Driver Name: com.mysql.jdbc.Driver
JDBC Url: jdbc:mysql://[serverName]/[databaseName]
Driver Available from: <http://www.mysql.com>

Database: PostgreSQL
Driver Name: org.postgresql.Driver
JDBC Url: jdbc:postgresql://[host]/[databaseName]
Driver Available from: <http://www.postgresql.org>

Database: DB2/AS400
Driver Name: com.ibm.as400.access.AS400JDBCdriver
JDBC Url: jdbc:as400://[serverName];translate binary=true
Driver Available from: jtopen project, the jt400.jar archive is the driver

Database: SQLServer (using jtds driver)
Driver Name: net.sourceforge.jtds.jdbc.Driver
JDBC Url: jdbc:jtds:sqlserver://[serverName]/[databaseName]
Driver Available from: <http://jtds.sourceforge.net>
Notes: Microsoft provides a driver for SQLServer 2000 and upper

Database: HSQLDB
Driver Name: org.hsqldb.jdbcDriver
JDBC Url: jdbc:hsqldb:fileName
Driver Available from: <http://hsqldb.sourceforge.net>

Database: Generic ODBC Datasource
Driver Name: sun.jdbc.odbc.JdbcOdbcDriver
JDBC Url: jdbc:odbc:<data-source-name>
Notes: The JDBC-Odbc bridge driver is available by default, you need to setup the DSN on the server (for optimal performance you should use the DBMS JDBC driver)



3.Groups

Groups are used to logically group queries. When a user logs in, he's asked to choose the group where to find the desired query. From the ART Admin Console follow the Groups option:

Manage Query Groups	
Add/Modify/Delete query Groups	
Group	General Ledger Graphs LIST Purchasing Update
Action:	ADD
<input type="button" value="Submit"/>	

At minimum, you need two groups:

(1) A **List of Values** (LOV) group: this is for queries used to retrieve list of values in order to prompt - in the parameters page, just prior to execute a query - a pop-up menu with the values. This special group - named "LIST" or "LoV"- should be present by default (it is created by the ART_tables.sql script during ART installation).

Note: If the "LIST" group is not present, the first group you define is automatically considered the List one. A LOV query does not have parameters.

(2) One or more **Generic** groups used to store the "real" queries or other objects.

A LOV query can have one or two columns in its resultset: the first is used to match the parameter, the second (if available) is the one displayed on the parameter page. Thanks to this, you can define a LOV query like:

```
SELECT location_id, location_name FROM LOCATIONS
```

the first column is the parameter to be passed to the "main" query: users see the location_name column while the match is performed using the location_id column.

You can modify the name or create new groups at any moment. In a standard organization, you may want to define several groups that match the different business areas of the company. For example:
INVENTORY, PURCHASING, FINANCE, PAYABLES, RECEIVABLES.

In order to create Development, QA and Production environments, it is common to create two or three groups (for example INV_PROD, INV_QA, and INV_DEV) and have a new developed query created in the %_DEV group, moved to (or copied and assigned to) the %_QA group for user functional tests and finally to %_PROD.

It is possible to limit the groups and datasources an administrator can deal with: thanks to this, there can be administrators that can act only in development environment and other ones that can deal with the "migration" process to production.



4. Users

4.1. Setup an User

ART always uses the connection parameters defined in the ART properties (see ART Installation Manual) and in Manage Databases in order to connect to databases.

Manage User	
Username	jacksf
Password	*****
Status	Active Disabled
Nick Name	Jack
Location	Venice
Country	IT
e-mail	jacksf@mycompany.com
Full Name	
Admin Level	Normal User Normal User allowed to schedule jobs Junior Admin (Query only)
COMMIT	

Note: if you use external authentication, you can leave the password field blank.
If you want a query to be executable by anyone without authentication, you must define a user with the special name `public_user` and grant to it the query. You need to setup a link to execute the query (magic link).
An administrator can create/update only administrators as powerful as himself.

When adding a new ART user, you must specify a password. If you are using an external authentication mechanism, you may leave the password field blank (or you may want to specify a password in order to give to a user the ability to log in both as internal or external).

The user password is stored in the ART repository in an encrypted form. The usernames are case insensitive, the passwords are case sensitive. Internal authenticated users can change their password once logged in.

An ART user can be promoted as an administrator by changing the **Admin Level** attribute. The levels are:

- **User:**
Normal User, Normal user allowed to schedule jobs
- **Low-priviledged administrators:**
Junior Admin (edit only queries), Mid Admin (can manage user privileges too)
- **High-priviledged administrators:**
Admin (can edit queries and users), Senior Admin (edit target databases, groups, rules, etc.), Super Admin (all).

It is possible to limit the datasources and groups a Junior or Mid administrator can deal with (see "Admin Privileges").



It is always possible to log in ART as a Super Admin by specifying the ART Repository username and password: if the ART Repository account gets expired or the ART Repository database is moved to another server and you need to change the url, this is the only way to log in and change the properties.

4.2. Setup the Public User

ART can be used to embed live data on web pages without user authentication.

If you want a query to be executable by anyone without authentication, you can define a user with the special name `public_user` and grant to it the query execution privilege.

In the Object Editor page is available the “direct” URL to execute the query/access the object: if the object has been granted to the public user, anyone can use the link.

This is useful to create web pages to display dynamic data without security restrictions.

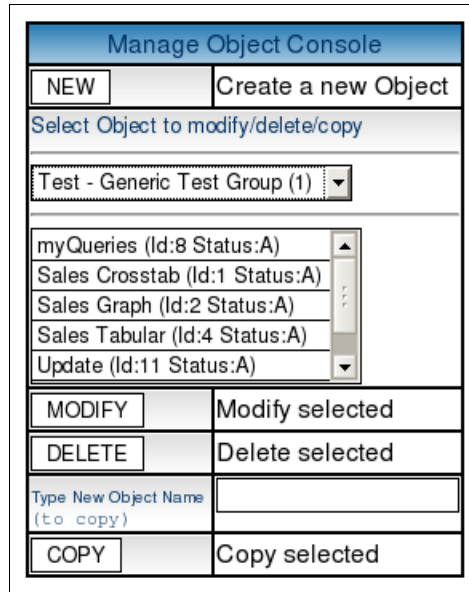
Note: Text Objects are public by default, all objects grouped within a Portlets Container need to be granted to the public user in order to have the Portlets Containers link working without authentication.



5. Queries

5.1. Introduction

From the ART Admin Console follow the "Objects" option to reach the **Manage Objects Console**. Here is possible to create/modify/update a query, or create a copy (clone) of an existent one:



An ART query is composed by three parts:

1. Header and Prepared Statement (SQL)

The header contains query properties like name, group, description, target database, type etc. and SQL (Structured Query Language) text.

2. Parameters

Before executing a query, users are prompted to enter values for query parameters. The values can be strings, integers, numbers, dates or can be picked from a pop-up list. The latter can have a single value or a pool of values (multi) and can be context sensitive - see Chained Parameters.

There are two types of parameters.

- **Inline** parameters are labels in the SQL query that are substituted with the parameter value specified by the user (label format is #name#).
Inline can be used to build Dynamic SQL code (see next chapter)
- **Multi** parameters are used to add a 'AND column in (<list of values>)' condition in the WHERE part of the query (users can select multiple values from a LOV query).

Inline and Multi parameters can be used to build "chained" values, i.e. the values displayed in a parameter drop down list depends on another parameter value.



3. Rules (optional)

Rules are used to filter the query result sets, to return only the rows where a column value matches a list of values associated to the user running the query. See the Rule section for more info. As in the multi parameters, the string 'AND rule_column_name in (<list of rule values>)' will be added at runtime to the SQL.

5.2. Header

From the Manage Object Console click on the "New" button to reach the query **Header** page. This page allows to specify the query name, description, target database, the Prepared Statement (SQL) etc.

Header	
Object Id	Auto
Group Id	4-Art Repository
Name	Art Objects
Status	Active
Short Description <small>(also: Title on Graphs/Portlets Container)</small>	Show info on Art Objects
Description <small>Max 2000 chars (also: Bottom Label on Graphs)</small>	Display Art queries names, target database, group and other information
Contact Person	jacksf
Uses Rules	No
Type	Tabular
Target Database	artRep3
SQL Statement source	
<pre>SELECT aq.NAME "Query Name" , aq.ART_FLAG "Status Active/Disabled" , aq.QUERY_ID " Id " , aq.SHORT_DESCRIPTION "Short Description" , aq.DESCRPTION "Description" , aq.SMART_RULES_FLAG "Uses Rules" , ag.NAME "Group Name" , ad.NAME "Database Name" FROM ART_QUERIES aq , ART_DATABASES ad , ART_QUERY_GROUPS ag WHERE aq.QUERY_GROUP_ID = ag.QUERY_GROUP_ID AND aq.DATABASE_ID = ad.DATABASE_ID AND aq.NAME like #query_name# AND ag.NAME like #group_name# AND ad.NAME like #datasource_name# ORDER BY aq.NAME</pre>	
<input type="button" value="Commit"/>	



Attribute	Description
Group id	Name of the group where you want the query to be located. The first available group is the "LIST" one (List Of Values-LOV) used to store queries used only to build list of values (for parameters).
Query Name	Name of the query as it appears to users. Do not use special chars in the name since it is used to build the export filename.
Status	If set to Inactive, the query will not appear to users
Short Description	Query short description. On graphs, this is the label appearing at the top (it can contain info on the graph size/color see below)
Description	Query description. On graphs, this is the label appearing at the bottom
Contact Person	Used to maintain the query contact person (it's just a label)
Use Rules	This flag specifies if the query uses rules: if set to yes, in the next page a button named "rule" will appear allowing to specify the column where to apply the rule (rule_column_name) and the rule name that will be dynamically applied to the query (see rule chapter for more information).
Query Type	See below for a detailed description
Target Database	It shows the available data sources. The query will be execute against the selected database.
SQL Source	A SQL query where parameters values are substituted with labels (#label# for inline parameters). Xml-style elements can be used to create Dynamic SQL statements (see next chapter). Some special tags can be used (:USERNAME, :DATE, :TIMESTAMP) – see hints on the Query Header pages.

Query Types

Tabular - Normal Query:

this is the default. A tabular result set exportable to a spreadsheet/pdf.

Tabular (html only):

a tabular result set that can only be displayed in html format. Useful to embed html code in the SQL query in order to change colors/size etc.

Crosstab:

produces a crosstab (pivot table) output exportable to a spreadsheet/pdf

The SQL resultset is expected to have 3 or 5 columns:

```
SELECT Dim1 "Label1", Dim2 "Label2", Value FROM ...
(data type: string, string, any)
```

or:

```
SELECT Dim1, AltSort1, Dim2, AltSort2 Value FROM ...
(data type: string, ... string, any)
```

The AltSort columns are used to sort the 1st column and 1st row values – for example, Apr-2006 is alphabetically before Jan-2006, the alternative sort column could be set to YYYY-MM format to order the period names in the right way.

Crosstab (html only):

same as crosstab but limited to html only.



Graph¹:

chart the result set (user can export the chart in pdf format).
The layout of the SQL must follow the below rules:

XY
SELECT ValueX "label X", ValueY "label Y" FROM ...
(data type: number, number)

Pie
SELECT Name "Label", Value FROM ...
(data type: string, number)

Bars / Stacked Bars / Line (category)
Static Series on columns
SELECT Item "Label Y", Value1 "SeriesName1" [, Value2, ...] FROM ...
(data type: string, number [, number, ...])

Dynamic Series
SELECT Item "Label Y", SeriesName, Value FROM ...
(data type: string, string, number)

Example:
SELECT Product "Item Label", Region, SUM(VOLUME) FROM...

Vertical Bar 3D

Item Label	East	North	South	West
Camera	1750	1600	1550	1500
Desktop	900	1200	1050	1250
Laptop	700	750	650	700

Stacked Vertical Bar

Item Label	East	North	South	West	Total
Camera	1750	1600	1550	1500	6400
Desktop	900	1200	1050	1250	4400
Laptop	700	750	650	700	2800

Line

Item Label	East	North	South	West
Camera	1750	1600	1550	1500
Desktop	900	1200	1050	1250
Laptop	700	750	650	700

Horizontal Bar 3D

Item Label	East	North	South	West
Camera	1750	1600	1550	1500
Desktop	900	1200	1050	1250
Laptop	700	750	650	700

1 To create graphs ART uses cewolf/jfreechart libraries that in turn use standard java AWT to plot charts: in order to work correctly AWT needs a graphic display. If you are using a "headless" workstation (i.e. a Unix box without X) you need to start the JVM with the option -Djava.awt.headless=true (JRE >=1.4).



Date/Time Line Series

Static Series on columns

```
SELECT Timestamp|Date "Label Y", Value1 "SeriesName1" [, Value2, ...] FROM ...
      (data type: timestamp|date, number, [, number, ...]). Date must be unique.
```

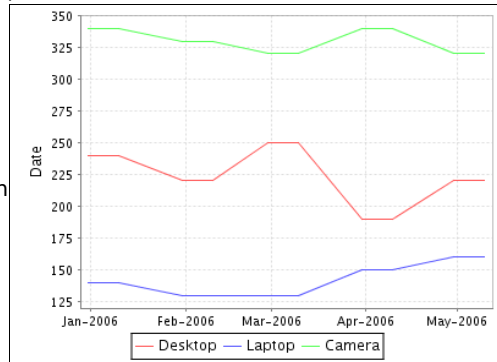
Dynamic Series

```
SELECT Timestamp|Date "Label Y", SeriesName, Value "Label Y" FROM ...
      (data type: timestamp|date, string, number)
```

Example:

```
SELECT ORDER_DATE "Date"
       , PRODUCT
       , SUM(VOLUME)
FROM ...
```

Note: in some RDBMS the DATE datatype is in fact a Timestamp.



Other Graph Settings

Use the query Short Description and Description to set a Graph Titel and bottom label. Default graph settings can be changed by appending to the Short Description a string with the following syntax:

```
@[WxH] [y1:y2] [nolabels] [nolegend] [#abcdef]
```

Examples:

```
@600x600 will produce a graph 600x600 (pixel)
```

```
@800x600 -10 +30 nolegend #FFFFFF will produce a graph 800x600 (pixel) focused on the y axis range [-10, +30] with a white background (#FFFFFF) and without the legend.
```

Report on Column #:

split the retrieved rows in order to group values. If the value is set to *n*, the query must be ordered by the the first *n-1* columns.

For example, if a query tabular output looks like:

AA	AA	B	C	D
AA	AA	E	F	G
AA	BB	H	J	K
AA	BB	X	Y	Z

using "Report on Column 2", the output would look like:

AA	AA	
B	C	D
E	F	G
AA	BB	
H	J	K
X	Y	Z

Update Statement:

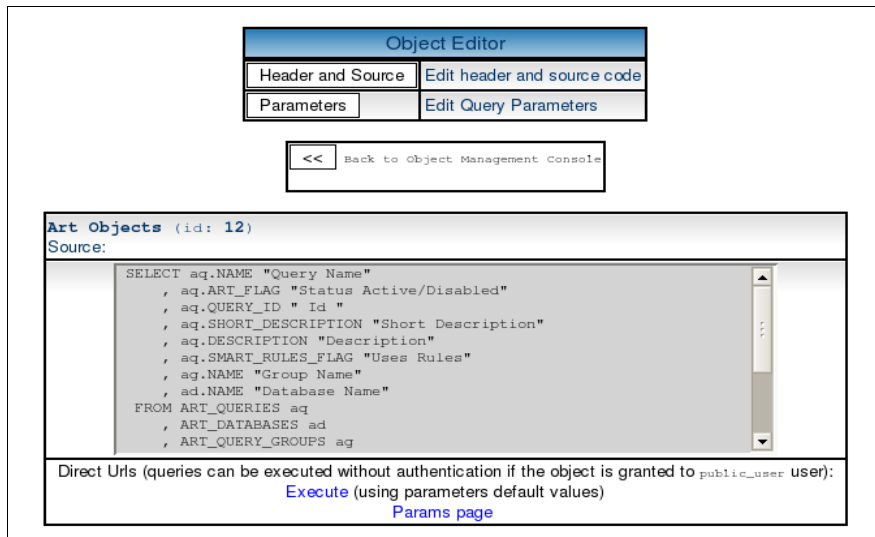
used to execute statements that do not returns any rows (INSERT/UPDATE/DELETE) or launch databae stored procedures.

Portlets Container / Text :

see chapters below



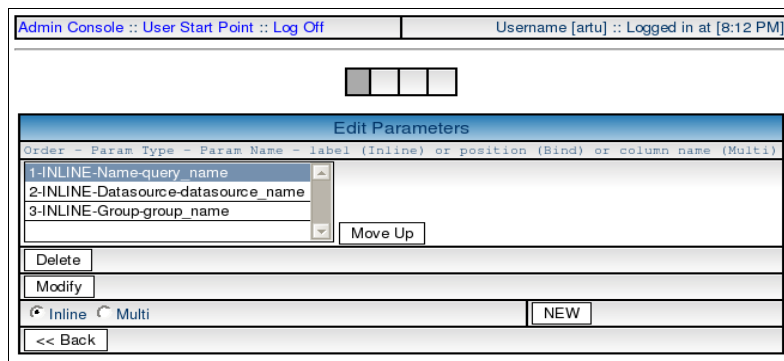
Once the Header&SQL page is complete, you can reach the Object Editor page (this is the same page you reach when modifying existing query objects):



Below the Object source are the links to execute the query directly (see [5.4])

5.3. Parameters

Once the Header&SQL is defined, click on the "Parameters" button to reach the "Edit Parameters" page:



Inline

An Inline parameter is a "label" in the SQL query, surrounded by the # char. For example, the following query has two inline parameters:

```
SELECT *
FROM orders
WHERE order_date > #date#
AND vendor_name like #vendor#
```



If the parameter is to be typed by the user, you should specify the type (integer, varchar, date...) and a default value (for dates use YYYY-MM-DD format). The type will be used to validate the user input.

Define Parameter	
Inline Parameter	
Parameter label <small>(without #, case sensitive)</small>	date
Name (User Viewable)	Order Date
Short Description	Order Creation date
Help Description	...
Class	DATE
Default Value <small>(For dates use YYYY-MM-DD format, default is today)</small>	2006-02-02
Use List Of Values (LOV) <small>(Set to Yes if the parameter should be picked up from a list)</small>	No
LOV query generator	
Apply Rules on LOV <small>(filter the list of values if a rule exist for the LOV query)</small>	No
<input type="button" value="Submit"/>	

If the parameter is to be obtained from a list (List of values – LOV), set to yes the "Use List Of Values" flag and select the LOV query generator (i.e. a query in the LIST group). If the LOV query uses rules, you may specify if the rule has to be applied to the LOV query.

```
ART uses the JDBC driver to pass Inline parameters values to the DBMS, converting them to the appropriate type (string, integer, number, date etc).  
  
SELECT *  
FROM atable  
WHERE date_column = #mydate# -- you do not need to convert mydate to a date object  
AND char_column like #myname# -- you do not need to put the ' char around the label  
  
Inline parameters can be used in Dynamic SQL and can be re-used in the same query (all their occurrences in the query are substituted with the value).
```

The same parameter (#label#) can be used multiple times in the same query: all occurrences will be substituted with the value.

Bind

Note: Bind parameters are un supported use Inline parameters instead. A query cannot have both bind&inline parameters.

Bind parameters match the ? char in the SQL. Their definition is similar to the inline parameters but you have to specify to which ? they match.

Multi

Multi parameters are used to create a drop down menu where users can specify more than one value (picked up from a List Of Values).



You need to specify the table column name (table_column)² that matches the values selected by the user: the string 'AND table_column in (<list>)' will be added to the prepared statement (SQL) just before executing it. The list of values is provided by one of the available LOV queries (the option ALL will appear on the user page to select all items – i.e. skip the additional condition).

The image below shows a parameter (Sales Region) where the available values are retrieved from the Regions LOV query and will match the REGION column:

Define Parameter	
Multi Parameter	
Table column	REGION
Name (User Viewable)	Region
Short Description	Sales regions
Help Description	Select one or more regions from the list (user CTRL+click)
LOV query generator	2 - Regions
Apply Rules on LOV <small>(filter the list of values if a rule exist for the LOV query)</small>	Yes No
<input type="button" value="Submit"/>	

For example, if the LOV queries shows the values North, West, East and South and the user selects the first two values, ART builds the string:
AND REGION in ('North', 'West')
and adds it to the WHERE part of the query (if your query has ORDER BY / GROUP BY make sure to type them in uppercase; ART will add the condition just after the last uppercase order by or group by).

² You might need to specify the column name with table alias (mytable.column_name) if the same column name is available in more than one table used in the SQL code.



5.4. Execute a query via URL

It is possible to execute a query assigned to the "public_user" (see [4.2]) via an ad-hoc URL without authentication. The view mode and parameters can be passed through the URL (all is case sensitive, special chars need to be substituted with the hex value).

In the Object Editor page, are the two links to:

1. Access directly the parameters page for the object
2. Execute directly the object (using default parameters values)

You may want to modify the 2nd link in order to override default settings, for example by changing the output mode from html to xls or change parameters.

Example 1: (inline)

```
http://server.name:port/ART/QueryExecute?QUERY_ID=120&QUERY_GROUP_ID=8&viewMode=html&P_startdate=2006-04-04&P_description=%25
```

executes query 120 (in group 8) in html mode; the #startdate# inline parameter (P_startdate) is set to 2006-04-04 and the #description# inline parameter (P_description) is set to % (%25 is the % char in hex). If you want the output as crosstab, you need to append the &_isCrosstab=Y value in the URL.

Example 2: (inline & graph)

```
http://server.name:port/ART/QueryExecute?QUERY_ID=120&QUERY_GROUP_ID=8&viewMode=graph&_GRAPH_ID=2&P_date=2006-04-04
```

_GRAPH_ID: 1 = "XY Chart", 2 = "Pie 3D", 3 = "Horizontal Bar 3D", 4 = "Vertical Bar 3D", 5 = "Line", 6 = "Time Series", 7 = "Date Series"

Example 3: (split report on column)

```
http://server.name:port/ART/QueryExecute?QUERY_GROUP_ID=2&QUERY_ID=122&viewMode=HTMLREPORT&SPLITCOL=2&P_param=abc
```

executes query 122 in report mode on column 2 (SPLITCOL); the first bind parameter is set to 'abc'.

Available View modes (**ViewMode**) are:

html, htmlGrid, htmlPlain, slkZip, xlsZip, xls, pdf, tsvGz, tsv, HTMLREPORT, xml, graph

It is not possible to pass "multi" parameters values via URL (any multi parameter will be ignored).

5.5. Rules

If the query uses rules, you need to specify the rule name and which is the table column upon which the rule will be built (rule_column_name). See chapter [12] for more information on rules.

5.6. An Example

This section shows how to define a simple query to retrieve the queries information (name, description etc) defined in the ART repository. There is only one parameter, obtained from a list of values: the query name.



In order to proceed with this example you need:

1. setup a datasource that points to the ART repository (you can use the same jdbc connection parameters you used in Art properties)
2. define a user and a query group (other than the LIST one)

List Of Values Query

This first query is the one used to retrieve the list of values (i.e. the list of available queries that will be shows as parameter).

From the ART Admin Console click the "Queries..." option and then on the "New" button.

Select the "LIST" group, set the query name to "ART Queries Names Lov" and type the following as Prepared Statement (SQL):

```
SELECT NAME FROM ART_QUERIES ORDER BY NAME
```

Make sure to leave the "use rules" attribute value set to "NO", select the ART repository database and click on the Submit Button. The query is created and the Object Editor page is now displayed.

Click on the Back To Query Management Console button to create a new query as specified in the next paragraph.

Main Query

This query retrieves the headers of the queries that are defined in ART, it has just one parameter whose values are retrieved from the "ART Queries Names Lov" query.

Following the same process used for the previous query, select a query group (not LIST one), name the query "ART Queries" and as SQL write:

```
SELECT NAME
      , SHORT_DESCRIPTION
      , DESCRIPTION
      , SMART_RULES_FLAG
      , UPDATE_DATE
FROM ART_QUERIES WHERE NAME = #query_name#
```

Make sure to leave the "Use Rules" flag value to "NO", select the ART repository as target database and click on the Submit Button. The query will be created and the Object Editor page will be displayed (see image in 5.2).

Now the "Parameter" button appears in the Object Editor page. Click on it and select "new" inline parameter (to substitute the #query_name# label at runtime). Type the label (without #, case sensitive), a friendly name, a short and help description. Set to "Yes" the "Use List Of Values" flag and select the "ART Queries Names Lov" as the statement to use to build the list (see image in 5.3). Then click "Submit".

Go back to the ART Admin Console and click on the "User Privileges" button, select an existing user and grant him the privilege to execute the query "ART Queries" and the appropriate group.

Now you can login as the user and execute the query:



ART Queries (ART at Sun Feb 22 19:50:27 CET 2004)			
NAME	SHORT_DESCRIPTION	DESCRIPTION	UPDATE_DATE
ART Queries	Show info on an ART Query	Shows headers information for ART Queris	2004-02-22
			Total Rows retrieved= 1

5.7. Other Useful Queries

It is useful to set up some queries against the ART Repository, in order to know the available queries, users , privileges etc. To do this a “target database” that points to the ART repository needs to be defined.

ART Queries: Show list of available objects in ART

```
SELECT aq.NAME "Query Name"
      , aq.ART_FLAG "Status Active/Disabled"
      , aq.QUERY_ID " Id "
      , aq.SHORT_DESCRIPTION "Short Description"
      , aq.DESCRPTION "Description"
      , aq.SMART_RULES_FLAG "Uses Rules"
      , ag.NAME "Group Name"
      , ad.NAME "Database Name"
FROM ART_QUERIES aq
      , ART_DATABASES ad
      , ART_QUERY_GROUPS ag
WHERE aq.QUERY_GROUP_ID = ag.QUERY_GROUP_ID
      AND aq.DATABASE_ID = ad.DATABASE_ID
      AND aq.NAME like #query_name#
      AND ag.NAME like #group_name#
      AND ad.NAME like #datasource_name#
ORDER BY aq.NAME
```

ART User Privileges: Shows who can execute an object

```
SELECT au.USERNAME "User Name"
      , au.ART_FLAG "Status A/D"
      , ag.NAME "Group Name"
      , aq.NAME "Query Name"
      , aq.QUERY_ID " Id "
      , aq.SHORT_DESCRIPTION "Short Description"
      , aq.DESCRPTION "Description"
FROM ART_QUERIES aq
      , ART_USERS au
      , ART_USER_QUERIES auq
      , ART_QUERY_GROUPS ag
WHERE aq.QUERY_GROUP_ID = ag.QUERY_GROUP_ID
      AND au.USERNAME = auq.USERNAME
      AND aq.QUERY_ID = auq.QUERY_ID
      AND aq.NAME like #query_name#
      AND ag.NAME like #group_name#
      AND au.USERNAME like #user_name#
      AND au.ART_FLAG like #active_flag#
ORDER BY 1, 2, 3
```



5.8. Limitations&Hints

1. You cannot create a query with only multi parameters and no other conditions (in the WHERE part). If the query uses rules, you cannot create it without condition (in the WHERE part). This is because, in both cases, ART adds the string "AND column_name IN (<list>)" at the end of the SQL (eventually before the ORDER BY or GROUP BY).

The workaround is to add a "dummy" condition if you need just multi parameters or rules. For example, you can add the string WHERE 1 = 1 at the end of your SQL.

2. XLS Export: ART uses the POI classes (jakarta/apache subproject) to create xls files. Those classes are optimized for creating complex xls files but not for "big" files - this is mainly due to the xls format: it is not possible to stream an xls file because the xls header can be created only when all the rows has been prepared! This means that the whole file need to be kept in memory. Depending on your hardware, you probably will obtain memory errors when creating sheets bigger than 10000/20000 rows. A maximum limit of rows is set in the web.xml to avoid this error (a warning message appears to the user). The workaround is to export large files in slk or tsv format.

3. Union/Subqueries with parameters

It is possible to use the UNION operator or subqueries with parameters if your database support them. The same parameter can be applied to all the SELECT statements:

```
SELECT A1 "A",B1 "B", ... FROM ...
WHERE A1 = #myparam#
UNION ALL
SELECT A2 "A",B2 "B", ... FROM ...
WHERE A1 = #myparam#
```

4. If you create an inline parameter - linked to a #label# - and afterwards you remove it, you need to manually update your SQL removing the extra inline label.

5. Three special tags can be used within the SQL and are substituted at runtime with their values:

:USERNAME replaced by the username of the user who is executing the query

:DATE replaced by the current date (format YYYY-MM-DD)

:TIMESTAMP replaced by the current time (format YYYY-MM-DD HH:MI:SS)



6. Dynamic Queries

It is possible to create Dynamic SQL queries using xml-like style tags in the SQL source code. This feature allows to modify the structure of the query based on user input.

The syntax (elements are case sensitive) is:

```
<IF>
<EXP1>value1</EXP1> <OP>operator</OP> <EXP2>value2</EXP2>
<TEXT> ... </TEXT>
<ELSETEXT> ... </ELSETEXT>
</IF>
```

Art parses the query and leaves only the text between the <TEXT> tag if the condition (*value1 operator value2*) is true or the text between the <ELSETEXT> tag if the condition is false.

The EXP1 or EXP2 content can be static values, inline parameters or :tags, while the OP content is an operator (see supported list below). Values are case insensitive.

For example in the following query:

```
SELECT *
FROM <IF><EXP1>#level#</EXP1><OP>equals</OP><EXP2>summary</EXP2>
    <TEXT> orders_summary </TEXT>
    <ELSETEXT> orders_details </ELSETEXT>
</IF>
WHERE VENDOR like #vendor#
    <IF><EXP1>#date#</EXP1><OP>is not null</OP>
    <TEXT> AND order_date > #date# </TEXT>
</IF>
```

if the #level# parameter is set to summary and the #date# one is not null ART rewrites the query as:

```
SELECT *
FROM orders_summary
WHERE VENDOR like #vendor#
AND order_date > #date#
```

if the #level# parameter is not set to summary and the #date# is null, ART rewrites the query as:

```
SELECT *
FROM orders_details
WHERE VENDOR like #vendor#
```

Operator <OP>	Description
eq or equals	= equals
la	< less than (alpha)
ga	> great than (alpha)
ln	< less than (numbers)
gn	> great than (numbers)
is null	applies to the EXP1 and returns true if the value is null
is not null	applies to the EXP1 and returns true if the value is not null
starts with	EXP1 begins with EXP2 string value
ends with	EXP1 ends with EXP2 string value
contains	EXP1 string contains in EXP2



Dynamic SQL examples

Dynamic OR/AND selection:

```
SELECT *
  FROM orders_summary
 WHERE VENDOR like #vendor#
    <IF><EXP1>#logic#</EXP1><OP>equals</OP><EXP2>OR</EXP2>
    <TEXT> OR country = #country# </TEXT>
    <ELSETEXT> AND country = #country# </ELSETEXT>
 </IF>
```

Dynamic ORDER BY:

The order by part is driven by the user input

```
SELECT *
  FROM orders_summary
 WHERE VENDOR like #vendor#
    <IF><EXP1>#sortby#</EXP1><OP>equals</OP><EXP2>Vendor</EXP2>
    <TEXT> ORDER BY 1 </TEXT>
    <ELSETEXT> ORDER BY 2 </ELSETEXT>
 </IF>
```

Dynamic query selection:

A different query is executed based on user input

```
<IF><EXP1>#showaddr#</EXP1><OP>equals</OP><EXP2>Y</EXP2>
 <TEXT>
   SELECT name, code, address, phone FROM VENDOR
   WHERE VENDOR like #vendor#
 </TEXT>
 <ELSETEXT>
   SELECT name, code, vat, pay_term FROM VENDOR
   WHERE VENDOR like #vendor#
 </ELSETEXT>
</IF>
```

Dynamic SQL with tags:

The condition is verified only if the date supplied by the user (#date#) is before the system date at query execution (:DATE tag)

```
SELECT *
  FROM orders_summary
 WHERE VENDOR like #vendor#
    <IF><EXP1>#date#</EXP1><OP>la</OP><EXP2>:DATE</EXP2>
    <TEXT> AND order_date > #date# </TEXT>
 </IF>
```

Check user input

Show a warning instead of execute the query

```
<IF><EXP1>#value#</EXP1><OP>ln</OP><EXP2>10000</EXP2>
 <TEXT> SELECT ... </TEXT>
 <ELSETEXT> SELECT 'Value too High' "Warning" </ELSETEXT>
</IF>
```

the select statement to use to show warnings depends from the DBMS (for example in Oracle you should use SELECT 'Value too High' "Warning" FROM DUAL)



7. Portlets Containers

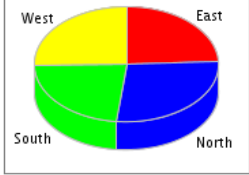
Define **Portlets Containers** to group queries, graphs or text objects in a single portal-like page:

Start :: My Jobs :: Log Off
Username [enrico] :: Logged in at [9:33 PM]

:: Portlets Container Example

Execute a graph


Sales by Area



■ East ■ North ■ South ■ West

Get a text object

This is a text fragment edited within ART (thanks to **Tiny CME** editor). Tiny offers a great **WYSIWYG** *Html Editor* where anyone can play with. It supports links ([ART Web Site](#))

and also images: 

text a text a text **a big text** a text a text A COURIER TEXT a text a text a text a text a text a text *an italic text* a text a text an underlined text a text in a justified paragraph

- Bullet1
- Bullet2

[\[edit\]](#)

Load from an external Url

This is a static web page!
Loaded from another source

Execute a crosstab query

VOLUME (PRODUCT_NAME / AREA)	East	North	South	West
Camera	310	340	300	320
Desktop	210	240	200	220
Notebook	110	140	100	120
Total Rows retrieved= 4				

Any Art query, graph or text object can be considered a portlet, i.e. a small embeddable frame within a web page.

Each portlet uses AJAX components to dynamically load itself within the page. Users can refresh or minimize each portlet independently by clicking on the buttons at the top of the portlet frame.

To define a new Portlets Container, select the “Portlet Container” object type in the Header page (see [5.2]): the page bottom part will slightly change and displays instructions. Instead of typing the source code of a query, an xml-like syntax can be used to define the Container structure.

Features:

- You can define as many columns as you need (tag: <COLUMN>)
- (optional) For each column you can specify a size (auto, small, ...) (tag: <SIZE>)
- Within each column you can specify one or more portlets (tag: <PORTLET>)
- Each portlet has a title (tag: <TITLE>) and can refer to an Art Object ID (tag: <OBJECTID>) to include an Art object like queries, graph, text or an external Url (tag: <URL>)

24/31



- (optional) Each portlet is rendered in the page and can be refreshed by the end user by clicking on the refresh button: you can set a refresh time (in seconds) to have the portlet to auto-refresh itself
(tag: <REFRESH>30</REFRESH>)
- (optional) Each portlet is rendered on page load: you can override this and let the user to decide if he wants to see the content (by clicking on the refresh button)
(tag: <ONLOAD>>false</ONLOAD>)

Syntax:

```
<PORTLETSCONTAINER>
<COLUMN>
<!-- column size: auto|small|medium|large default is auto-->
<SIZE>medium</SIZE>
<!-- create a new portlet within this column
to embed an art object (query, graph, text) -->
<PORTLET>
<TITLE>Portlet title</TITLE>
<!-- (optional, default is true) load content when page appears -->
<ONLOAD>>true</ONLOAD>
<!-- (optional, default is never) refresh content every 30 seconds-->
<REFRESH>30</REFRESH>
<!-- embed Art object -->
<OBJECTID>2</OBJECTID>
</PORTLET>
<!-- create a new portlet within this column
to embed an external html fragment -->
<PORTLET>
<TITLE>Portlet title</TITLE>
<URL>http://my.site.com/page.html</URL>
</PORTLET>
<!-- .. you can add as many portlets as you want -->
</COLUMN>
<COLUMN>
<!-- you can add as many columns as you want -->
</COLUMN>
</PORTLETSCONTAINER>
```

In the Object Editor page is the link you can use to access directly the Portlets Container. This link can be used by anyone (without authentication) but the portlet contents will be displayed only if the object have been granted to the public user.

When a user logs in to execute a portlet container, he can select if to modify the default parameters used by the queries. All the embedded queries are parsed and the parameters are displayed: if more queries uses the same inline parameter label, this is displayed only once.

Hints&Limitations

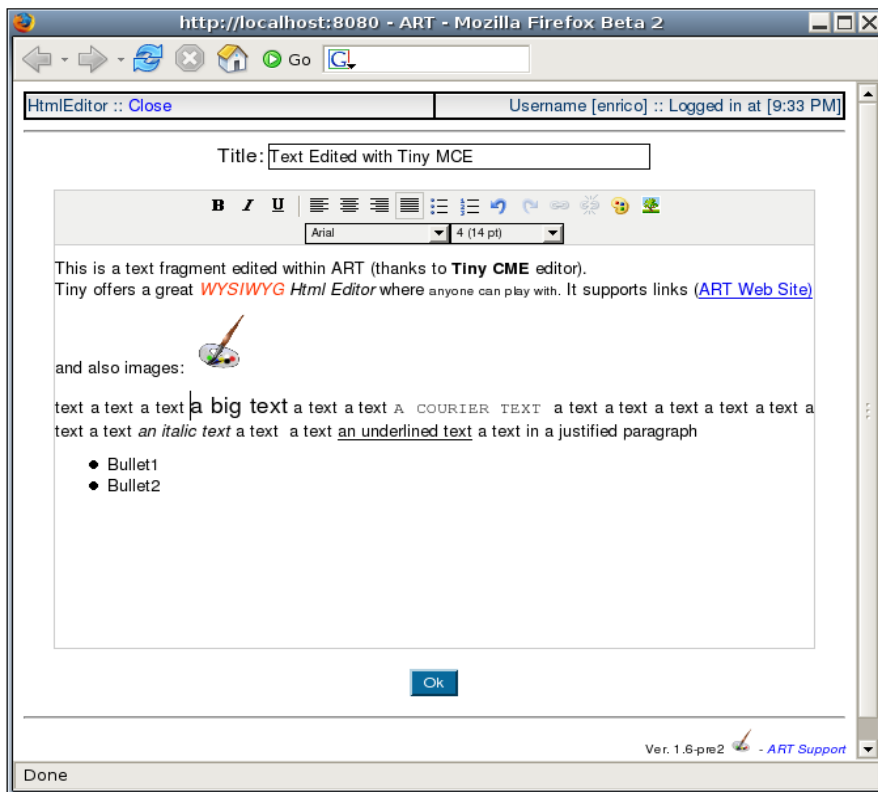
- Using the <OBJECTID> tag, tabular/crosstab queries are executed and displayed in html format. To override this, use the <URL> tag typing the URL as described in [5.4] (default parameters will be used)
- Tag names are case sensitive (use uppercase)
- Minimum refresh allowed is 5 seconds – pay attention to do not overload server and databases...
- Only in-line parameters can be updated: multi parameters are skipped



8. Text Objects

A Text Object is just an html text fragment that can be viewed as a portlet within a portlets container or as a standalone web page.

When a user is viewing a text object, if it is has been granted to him an edit link appears at the bottom in order to edit the content using a WYSIWYG editor emebded in Art:





9.Scheduling

In order to schedule a query (job) a user need to be setup as a “Normal user allowed to schedule jobs” (see [4.1]). The user e-mail needs to be defined correctly and a valid SMTP server needs to be specified on ART Properties.

Once logged in the ART User part, user's scheduled jobs are available following the My Jobs link on the top bar.

To schedule a new job, select the Group and the Query to schedule. In the Input Parameter page, specify the parameters and select the value “Schedule Job” from the “View Mode” list. Then click on the Execute button.

The next page appears allowing to specify when to run the job and the job type:

Schedule a new ArtJob for query: <i>CrossTab</i>		
Job Information		
Job Type	Output Type	Enable Auditing
Publish	Spreadsheet (zip xls)	Yes
From	enrico1@computer.org	
To (separate addresses with ';' char)	first@email.com; second@email.com	
Message	The Job has been executed	
Job Schedule		
Month	Day	WeekDay
Every Month	Every Day	Every Week Day
Hour	Minute	
Every Hour	30	
<input type="button" value="Schedule this Job"/>		
<small>To delete an existing job or to review info, you can follow the My Jobs link after the logon screen</small>		

Ver. 1.1.0 - [ART Web Site](#)

Job Type

- **E-Mail Output:** the output is emailed to specified recipients ('To' text area).
- **Publish:** the output is saved to a file, the file is reachable from the My Job pages. Once created, a notification mail is sent to specified recipients (leave the To area empty to do not send any mail). Published files are deleted after one day.
- **Alert:** send an e-mail to the specified recipients only if the first cell of the query result is greater than zero (typically you would use database functions to return a non zero value in case of a business event, in order to trigger an ad-hoc alert).
- **Just Run It:** simply execute the statement (useful to launch database procedures or perform periodic data updates).

Enable Auditing

Use this flag to enable/disable auditing information (start/end date) for the job. If set to no only the last start/end date is recorded. Otherwise the table ART_JOBS_AUDIT will contain a line for each job execution.



10. User Privileges

From the ART Admin Console, the "User Privileges" option allows to define which objects a user can execute and which groups he has access to. A user must be granted to both the object and the group in order to be able to select and execute the object in the group.

You can select more users at the same time (ctrl+click) and therefore apply the GRANT or REVOKE action to a set of users. If you grant an already granted query, a warning message appears.

Note: Text objects are public by default – i.e. any user could view them. If a text object is additionally granted to a user, the user will notice an edit link at the bottom of the text: this allows to update the text using the embedded html editor.

Note: it is possible to modify a context parameter in the WEB-INF/web.xml file in order to have the security based only on groups (a user will be able to execute all the queries in a group if he has been granted to it).

11. Admin Privileges

From the ART Admin Console, the "Admin Privileges" option allows to define which Groups and DataSources a low-privileged administrator (Junior- and Mid-) can deal with.

By default high-privileged administrators are limited only in functionalities (manage users, manage datasources etc) but can deal with queries in any groups or sourcing data from any database.

You can select more users at the same time (ctrl+click) and therefore apply the GRANT or REVOKE action to a set of administrators. If you grant an already granted datasource or query, a warning message will appear.



12. Rules

Security Rules are used to filter the query result sets. They allow to have the same query to be automatically filtered depending on the user that is executing it.

The security rules works as in this example:

- you have a table (employees) with the a column named "region"
- create a rule named "GeoArea"
- setup a query that selects rows from the table employees and link it to the rule named "GeoArea" on the column employees.region
- link the user to the rule "GeoArea" for values NORTH and EAST

When the user executes the query, he will extract only the rows where the values NORTH or EAST matches the column region (i.e. the SQL query will be modified on the fly before execution by adding the string `AND employees.region IN ('NORTH', 'EAST')` in the WHERE part).

The following steps have to be performed in order to use rules:

1. **Define the rule name** (it's just a label)
From ART Admin Console click on the "Rules" option and specify the rule name
2. **Link the query to the rule**
Set to yes the "Uses Rules" flag on the query Header and, from the ART Query Editor page, a new button allows to attach a rule to a column name
3. **Assign to users the rule values** they can deal with
From ART Admin Console, click on the "Users Rules" button, select the user and specify the rules values for the selected user.

Rules values can be EXACT values or LOOKUP to other users.

You may want to create a "dummy" user in order to group rule values: for example you can create one user with username "NorthEast" and assign to him - for the rule "GeoArea" - the values NORTH and EAST.

Then for every user you want to use such rule values, you can simply specify the "GeoArea" rule values as a lookup to the NorthEast user.

Note: if a query is linked to a rule but the user that tries to execute the query has not been setup for that rule, the query execution will not be performed and an error message will appear.



13.Chained Parameters

A parameter can be setup in order to retrieve the values from a LoV that is dynamically filtered depending on another parameter.
 The latter - called "master" - need to be an inline type.
 The driven parameter - called "slave" - need to be bound to a special LoV where the #filter# label is specified in the WHERE part of the SQL.
 By using dynamic SQL, any LoV can be converted to be used both as a normal Lov and a filtered Lov.

Usage Example

Table COUNTRIES has column COUNTRY_NAME
 Table CITIES has columns COUNTRY_NAME, CITY_NAME

Create the LoV used by the master parameter:	Create the LoV used by the slave parameter ³ :
<pre>SELECT COUNTRY_NAME FROM COUNTRIES ORDER BY 1</pre>	<pre>SELECT CITY_NAME FROM CITIES WHERE COUNTRY = #filter# ORDER BY 1</pre>

When you define a slave parameter, select the filtered LoV as the values generator and specify the position of the Master parameter that drives the filter:

Inline Parameter	
Parameter label (without #, case sensitive)	city_name
Name (User Viewable)	City
Short Description	Select City
Help Description	Values are dynamically build based on country
Type	VARCHAR
Default Value (For dates use YYYY-MM-DD format, default is today)	
Use List Of Values (LOV) (Set to Yes if the parameter should be pick'ed up from a list)	Yes No
LOV query generator	28 - Cities
Apply Rules on LOV (filter the list of values if a rule exist for the LOV query)	Yes No
Chained LoV (this param - slave - depends on values of another parameter - master)	Chain on param 1 ?

On the query parameter page, Art will automatically trigger the LoV query when the master parameter value changes, applying the filter (i.e. the master value is

³ Using dynamic SQL you can create a LoV parameter that can be used also as a non-chained one, in the following example the WHERE part is ignored if the #filter# value is not available:

```
SELECT CITY_NAME FROM CITIES
<IF><EXPI>#filter#</EXPI><OP>is not null</OP>
<TEXT>WHERE COUNTRY_NAME = #filter#</TEXT>
</IF>
ORDER BY 1
```



considered as an inline parameter named #filter#):

Sales Chained				
Name	Value	Type	Description	Help
Country	Japan	ABC	Select Country	...
City	Tokio	Country	Select City	...
View Mode	Browser (Fancy)			
<input type="checkbox"/> Show Parameters			Execute	

Note: the master parameter is expected to be an inline one but it does not need to be a LoV one; in this case the value typed by the user triggers the slave LoV when the master text field changes the focus (press Tab).

Note: if the slave values do not get populated, check the LoV query syntax / try to execute it as a normal Lov to debug the issue.

Note: Art allows to create an inline parameter only if the label is available in the mail query. If the main query does not have a column that matches the label use the dummy condition:

```
WHERE #master_label# = #master_label# AND ...
```

A slave inline parameter can be considered as a master for another parameter (this is why this is called chained...).

... check for updates on <http://art.sourceforge.net>, use the help forum for questions and enjoy ART!