xCelerator *XDQL* Installation and User Guide

Version 1.0

Preface

Purpose

This document provides step-by-step instructions on how to install and configure the xCP xCelerator XDQL.

Intended audience

The information in this guide is intended for xCP developers willing to leverage XML in DQL queries.

The objective of this xCelerator is to provide ability to retrieve DQL queries result as XML files, including contents of the objects included in the result set.

Support information

The xCP XDQL xCelerator is not supported by EMC.

Related documentation

The EMC Documentum xCelerator home page (~~https://community.emc.com/docs/DOC-6143~~) provides access to xCP xCelerators.

Revision history

Date	Version	Change
25/10/12	2.0	Migration for XCP 2.0

Contents

Overview

What is an xCelerator?

An xCelerator is one or more artifacts that can be used to accelerate the creation, adoption, and/or implementation of an EMC Documentum xCelerated Composition Platform (xCP) solution. An xCelerator is not necessarily a running application; instead, it is intended to hasten application development by providing key pieces of functionality out-of-the-box. Examples of this include:

- A sample business process flow created by Process Builder
- A sample high fidelity form using Forms Builder
- A sample Business Activity Monitor (BAM) dashboard
- A sample bit of code or configuration that connects the workflow with a third-party product
- Documentation that focuses on best practices and troubleshooting when developing and implementing an xCP solution
- A sample application

xCelerators are:

- Technology focused and can be samples, examples, starting points, or guides that accelerate the creation of an xCP solution
- Free and are not EMC-supported products, yet they do go through a formal release cycle, including quality assurance and documentation
- EMC intellectual property
- Sharable with multiple partners or anyone who wants to download them from the xCP community site

xCP XDQL xCelerator

Bill of Materials

The xCP XDQL xCelerator is accessible through the Community xCelerator xChange home page (~~https://community.emc.com/docs/DOC-7597~~) by selecting the xCP XDQL xCelerator download link.

The XDQL contains:

Artifact	Path / File Name
Overview guide (this document)	xcelerator-xdql.pdf
Installation files	Package/xcp2_xdql.jar

Source code	Source

Installing xCP XDQL xCelerator

Prerequisites

In addition to Documentum Content Server 7, the following must also be installed prior to installation of this xCelerator:

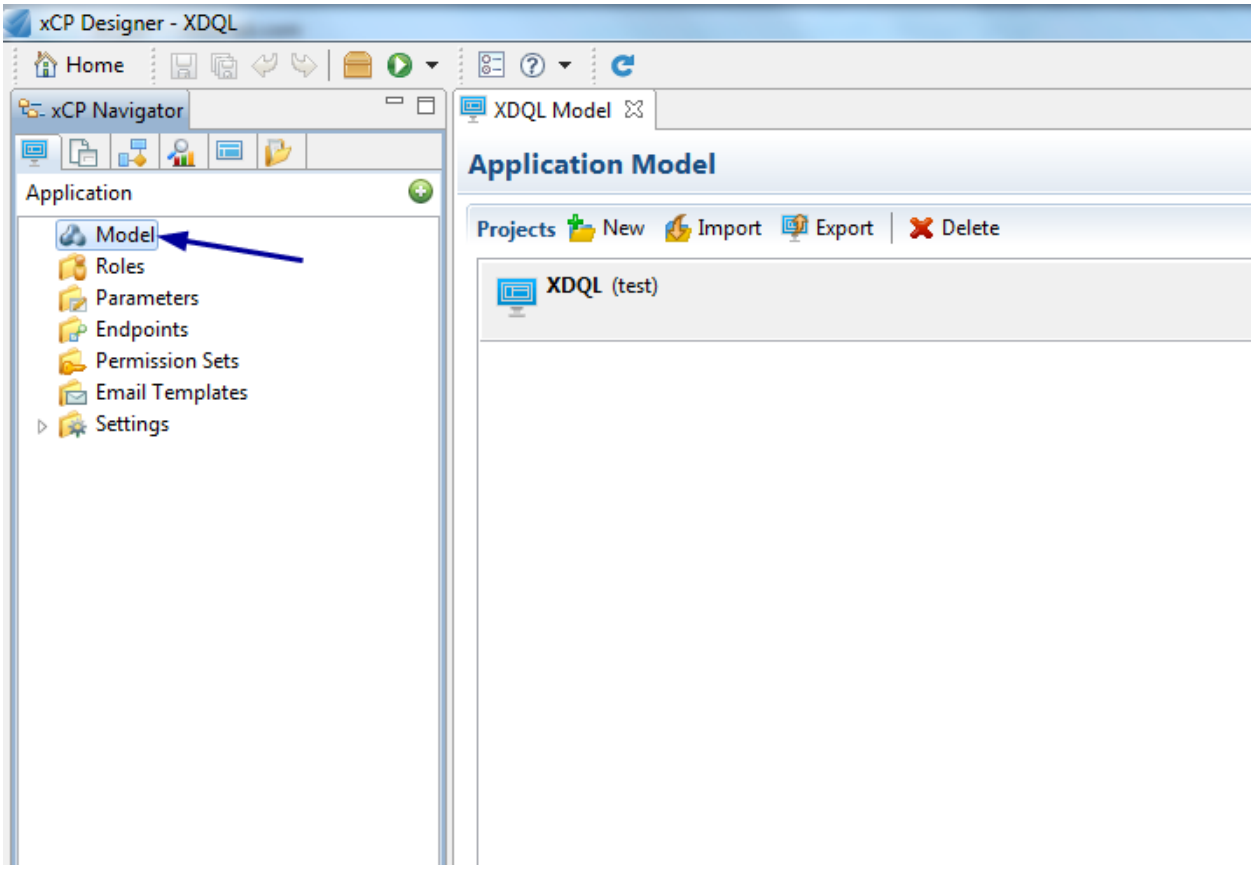
- Documentum XCP Designer 2.0

Install the XDQL xCelerator file

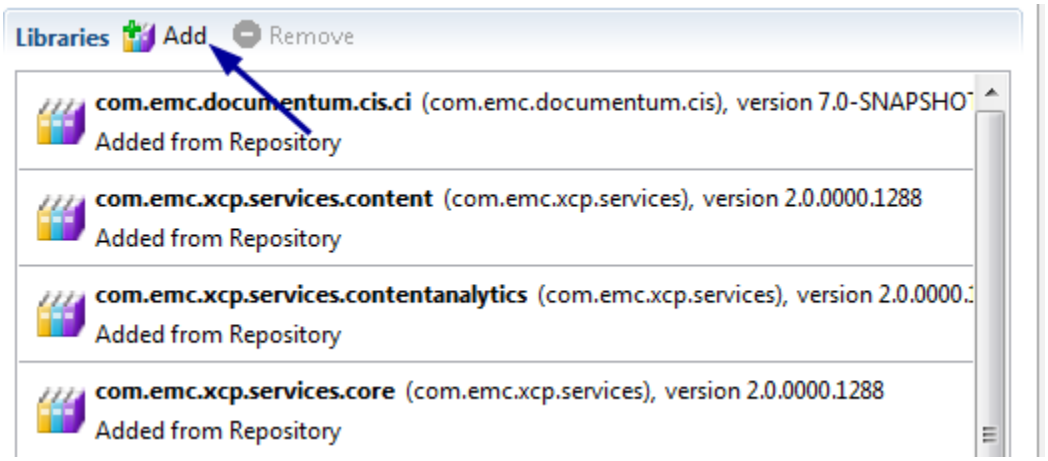
Download the xcp2_xdql xCelerator.zip file which is accessible through the EMC Documentum xCelerator community home page by selecting the XDQL xCelerator link.

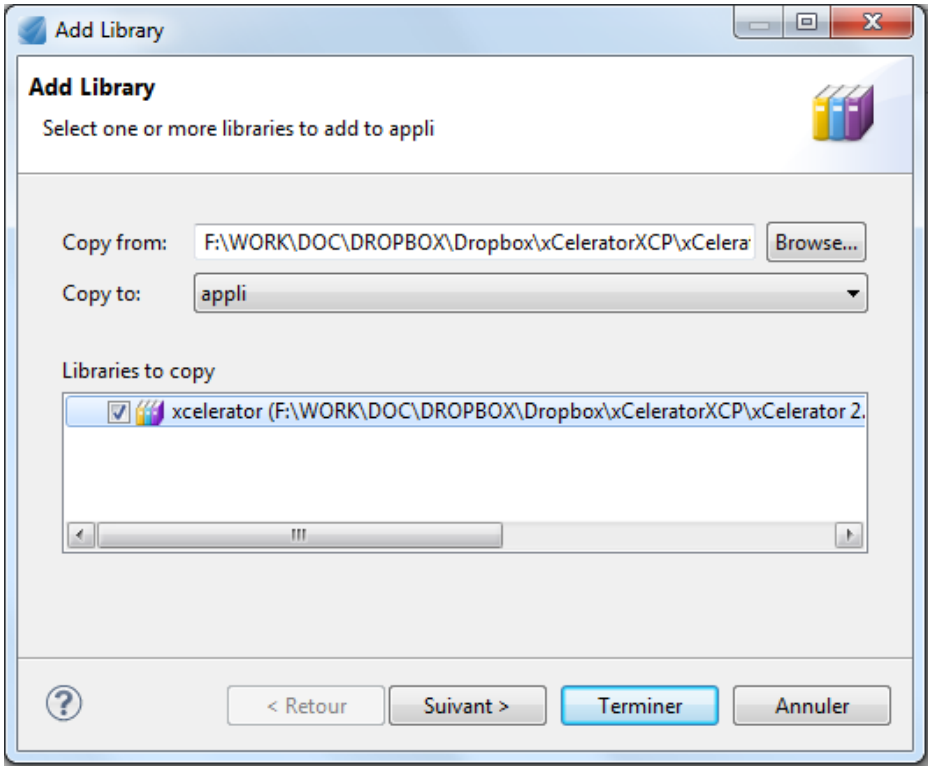
Unzip the xcp2_xdql xCelerator.zip file and locate the **xcp2_xdql.jar** at the following path: *<Install folder>/xcp2_xdql xCelerator/Package/xcp2_xdql.jar*

Open your application in xCP Designer and select “Model” in the “Application” tab

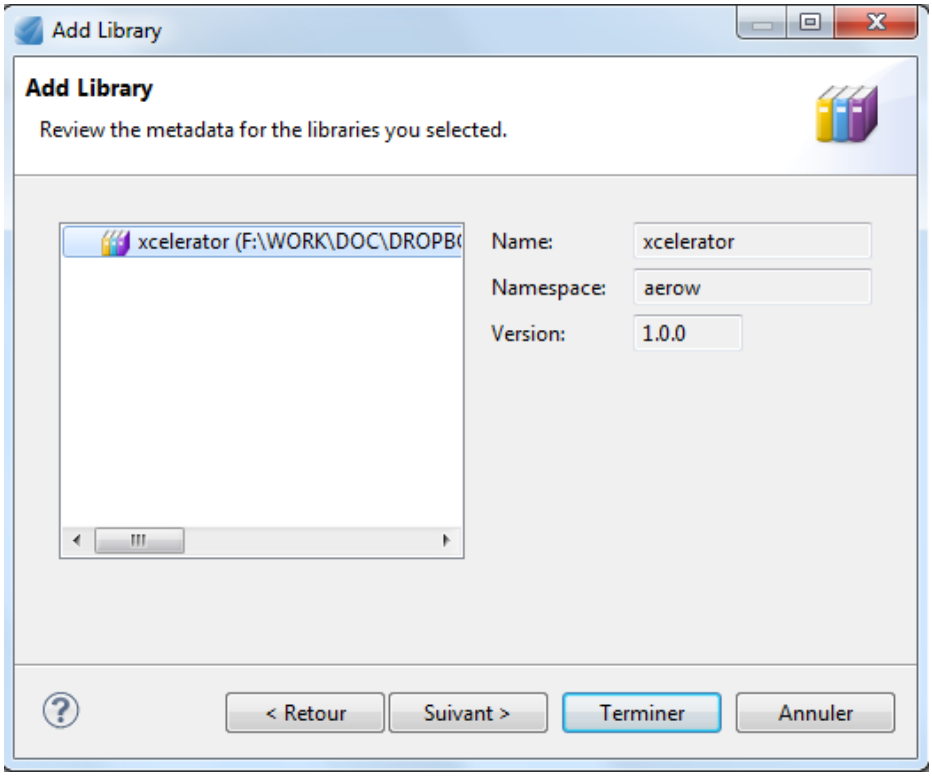


In the Libraries part, click on the “ADD” button. A pop-up will open. Click the “Browse” button and navigate to the folder *<Installed folder>/xcp2_xdql xCelerator/Package* and select **xcelerator** in “Libraries to copy”. Click on the Next Button





Check if the same name and the Namespace are well filled and click on the “Finish” Button


Add Library
Add Library
Review the metadata for the libraries you selected.
xcelerator (F:\WORK\DOC\DROPB
Name:
xcelerator
Namespace:
aerow
Version:
1.0.0
< Retour
Suivant >
Terminer
Annuler

Using the XDQL xCelerator

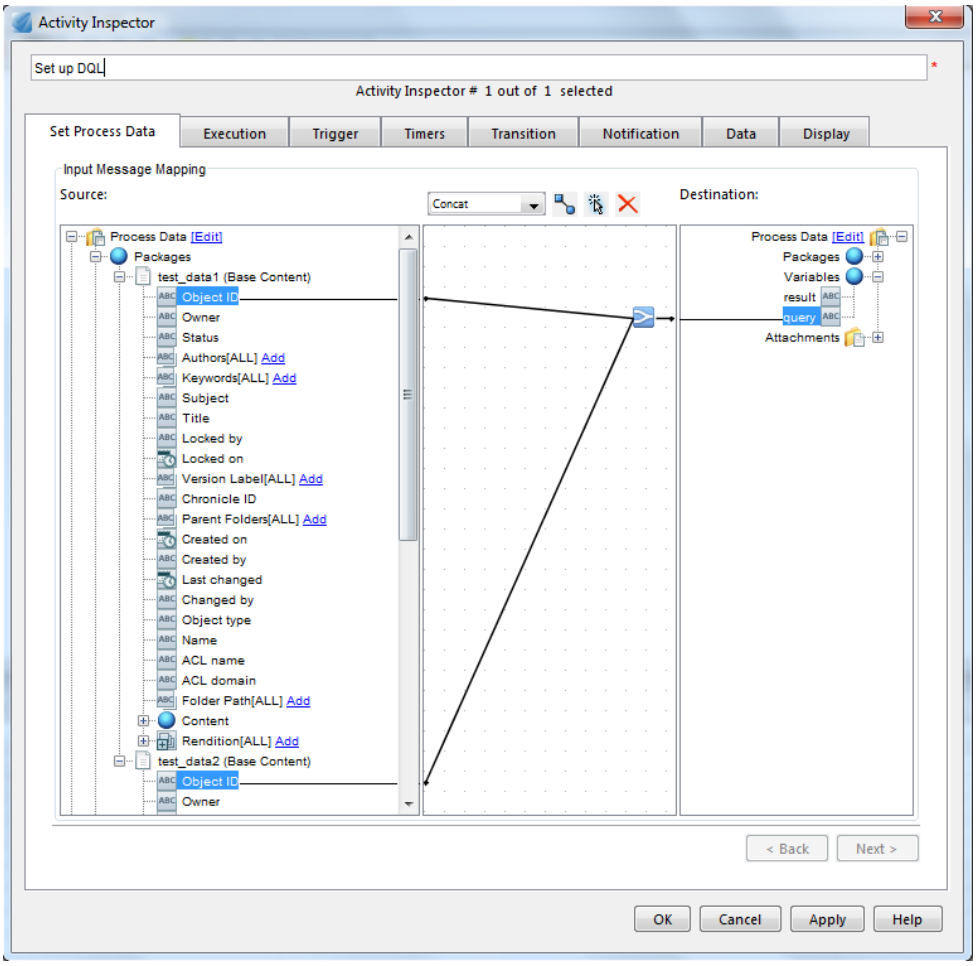
Configuration

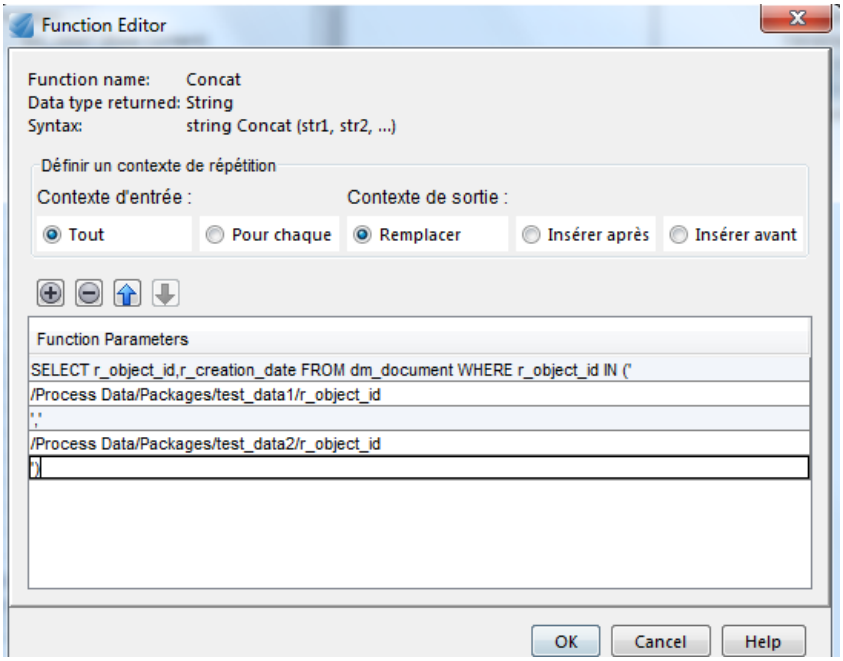
XDQL xCelerator is provided as a java extension. This makes the xCelerator available in all environments straight after the jar files and the java module file had been installed, no additional steps are required.

To execute DQL queries and retrieve result as XML the following steps are required.

- Set up the DQL query
- Use Process Data Mapping create DQL query and store it as a process variable. This is currently the only way to provide a DQL query to the XDQL xCelerator.

You can use Concat function for providing passing parameters to the query. The following configuration will generate query selecting objects with the given object IDs:





The output query will be the following:

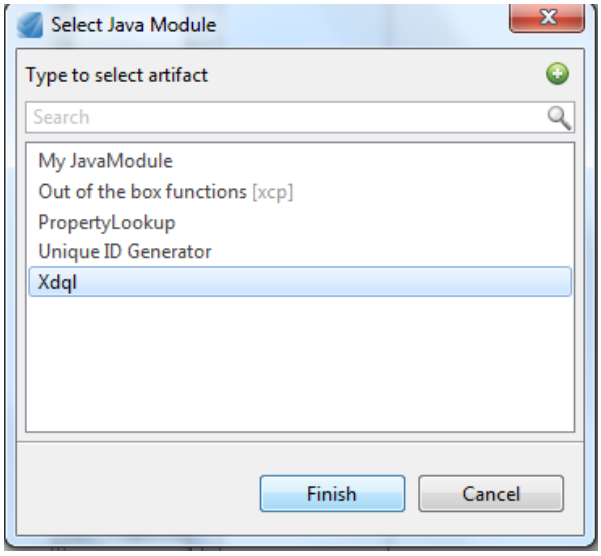
```
SELECT r_object_id, r_creation_date FROM dm_document

WHERE r_object_id IN ('id1', 'id2')
```

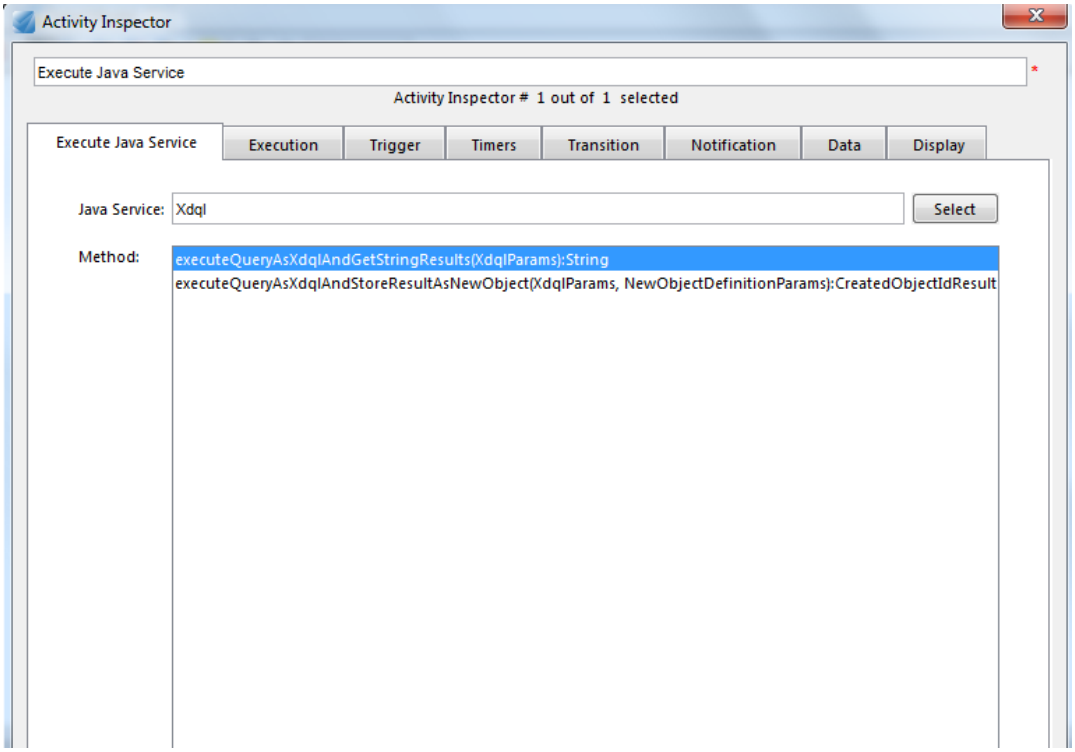
Where id1 and id2 will be substituted with the real values.

Configure XDQL xCelerator – common steps

1. Drag the “Execute Java Service” activity template to your process
2. In the activity inspector select Execute Java Service configuration tab
3. Click on “Select” button to select Java Module.



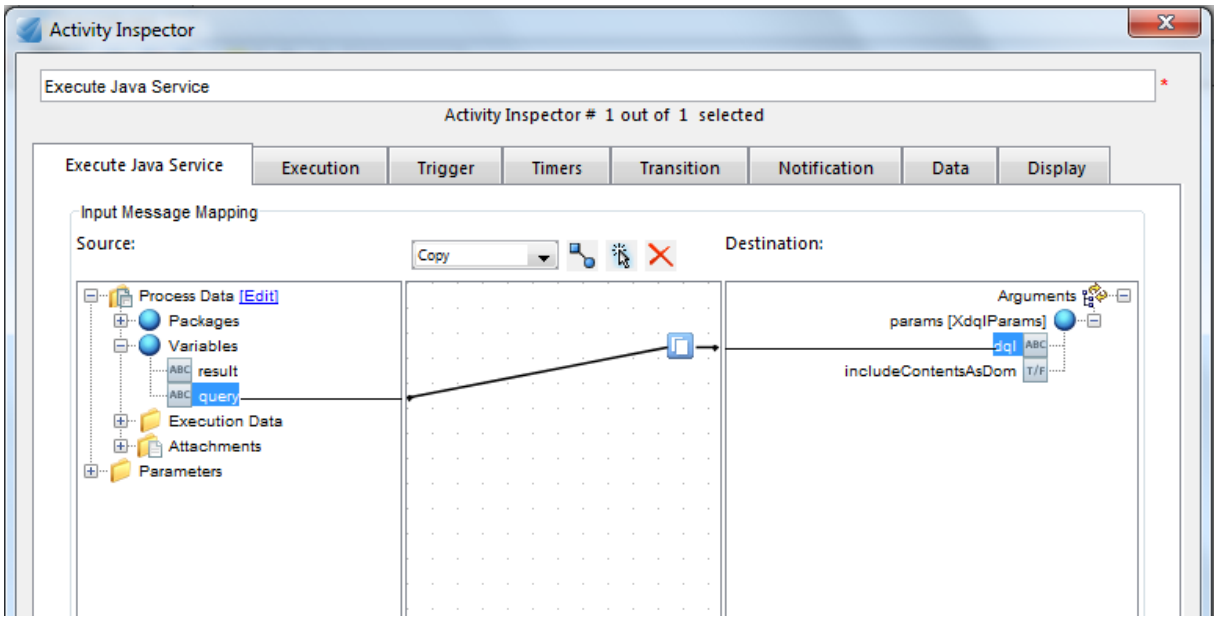
1. Select Xdql and click Finish.
2. Make sure that Xdql module is selected as module interface:



Configure DQL query

Regardless of the method being executed (either storing returning content as string, or creating new file), DQL query needs to be defined.

DQL query is passes in XdqlParams.dql argument, the following data mapping screenshot provides an example:



The optional includeContentsAsDom parameter specifies if content is to be included in the resulting XML. This is useful when querying for objects with XML content.

Example result of query being executed with includeContentsAsDom set to false:

```
<?xml version="1.0"?>

<root>

<object ID="0901e30680013261">

<r_object_id>0901e30680013261</r_object_id>

<r_creation_date>8/22/2010 5:40:48 PM</r_creation_date>

</object>

<object ID="0901e30680013262">

<r_object_id>0901e30680013262</r_object_id>

<r_creation_date>8/22/2010 5:40:48 PM</r_creation_date>

</object>

</root>
```

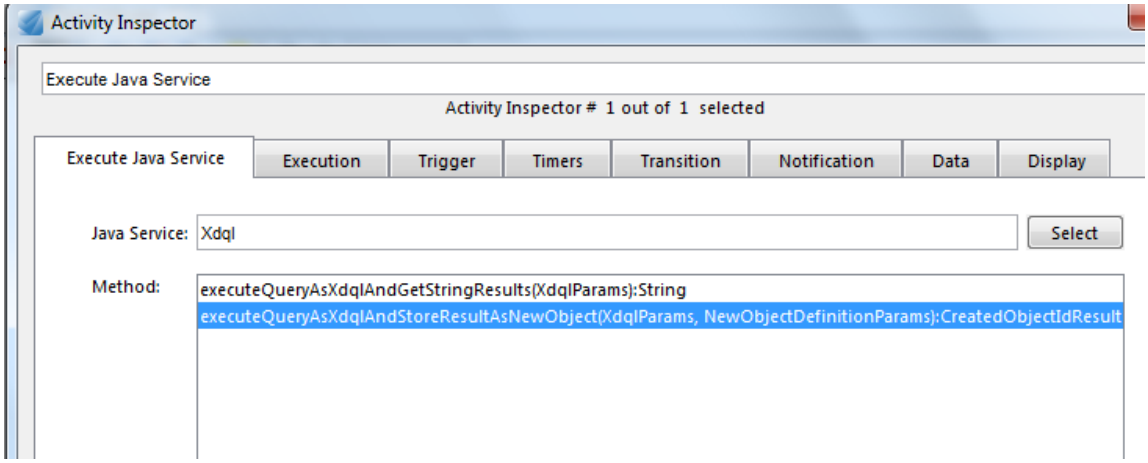
Example result of query being executed with includeContentsAsDom set to true:

```
<?xml version="1.0"?>
```

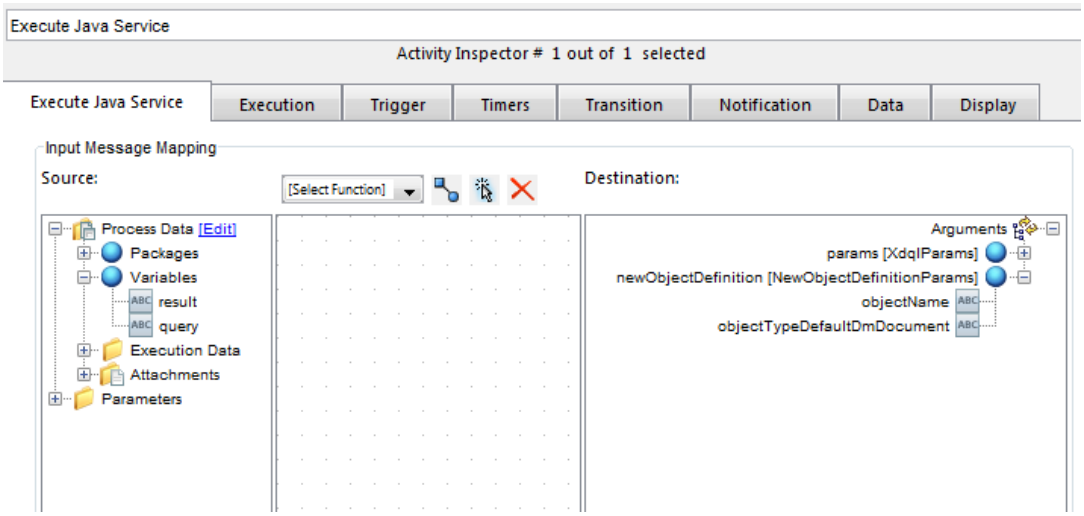
```
<root>
<object ID="0901e3068001325e">
<r_object_id>0901e3068001325e</r_object_id>
<r_creation_date>8/22/2010 5:39:17 PM</r_creation_date>
<content encoding="dom" mime-type="text/plain">
<ef_xdql_demo>
<sample_string>Sample string in document 1</sample_string>
<sample_date>2010-08-22T15:39:08Z</sample_date>
</ef_xdql_demo>
</content>
</object>
<object ID="0901e3068001325f">
<r_object_id>0901e3068001325f</r_object_id>
<r_creation_date>8/22/2010 5:39:15 PM</r_creation_date>
<content encoding="dom" mime-type="text/plain">
<ef_xdql_demo>
<sample_string>Sample string in document 2</sample_string>
<sample_date/>
</ef_xdql_demo>
</content>
</object>
</root>
```

Configure XDQL xCelerator to store create new object for the resulting XML

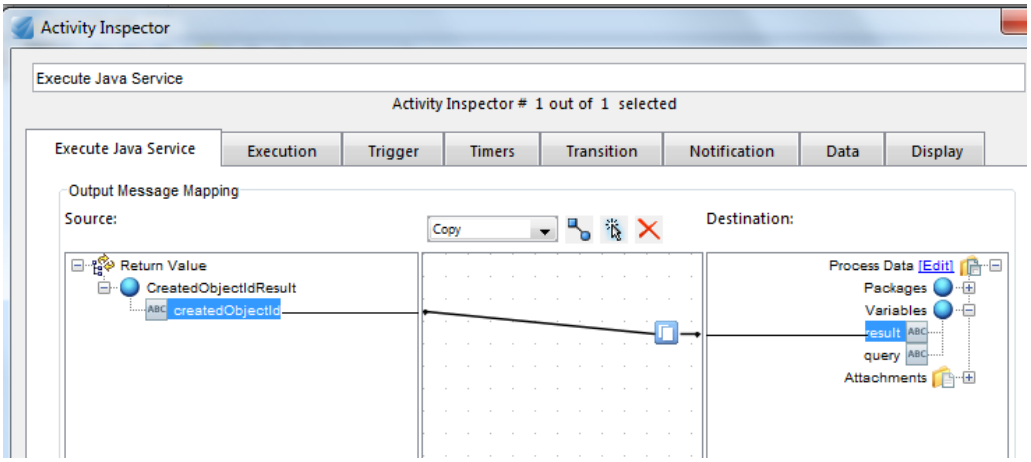
XDQL xCelerator will create new object for the resulting XML if executeQueryAsXdqlAndStoreResultAsNewObject is selected for execution:



Optional configuration include specifying object type of the object being created and its name. By default an instance of dm_document is created with name starting with xdqlresult_. These are configured by the *NewObjectDefinitionParams* argument:



Output mapping will let you store created object in the workflow's package:

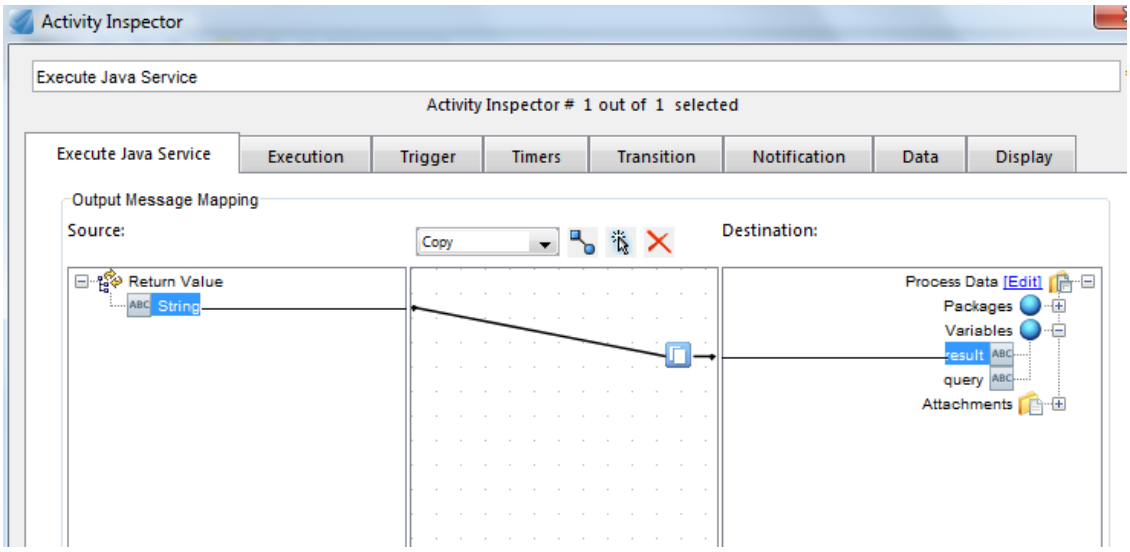


Configure XDQL xCelerator to store resulting XML in the process' variables

Use this option with cause as size of the XML file can easily exceed variable's length.

XDQL xCelerator will return the resulting XML as string if *executeQueryAsXdqlAndGetStringResults* is selected for execution:

Result can be simply mapped into process' variable:



Known issues

If query execution fails exception is not thrown.

Error message is included in the resulting XML.

The following query:

```
SELECT abcd FROM dm_document
```

Will produce the following XML output instead of stopping execution:

```
<?xml version="1.0"?>

<error>DfException:: THREAD: pool-1-thread-2; MSG:

[DM_QUERY_E_NOT_ATTRIBUTE]error: &quot;You have specified an

invalid attribute name (abcd).&quot;; ERRORCODE: 100; NEXT: null</error>
```

IncludeContentsAsDom set to true will cause an error when query returns dm_folder instances:

The following query:

```
SELECT r_object_id FROM dm_folder ENABLE (RETURN_TOP 1)
```

Will produce the following XML output instead of stopping execution:

```
<?xml version="1.0" ?>

<error>java.lang.NullPointerException</error>
```

The same query with includeContentsAsDom set to false gives valid output:

```
<?xml version="1.0"?>

<root>

<object ID="0b01e30680000108">

<r_object_id>0b01e30680000108</r_object_id>

</object>

</root>
```