xCelerator *PropertyLookup* 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 PropertyLookup.

Intended audience

The information in this guide is intended for PropertyLookup. The objective of this xCelerator is to provide Property lookup.

Support information

The xCP PropertyLookup 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	Author	Change
20110513	Draft	Ingemar Axelsson	Created
20121024	2.0	Aerow ECM World	Migration to 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 PropertyLookup xCelerator

The PropertyLookup xCelerator is used in a workflow to look up properties from a property file that resides in a repository. This can be used to implement customizations in a process based on predefined values stored in a simple java.util.Properties file.
PropertyLookup is intended to be used in process builder to allow a process to look up values based on given keys during a workflow.
There are many ways of using the PropertyLookup xCelerator, either by using property files known and created design time, or by retrieving the property files during runtime.

The PropertyLookup xCelerator uses a cache mechanism that is looking at the property files modification date to decide if a property file should be reloaded or if the properties in the cache could be used.

.

Bill of Materials

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

The PropertyLookup contains:

Artifact	Path / File Name
Overview guide (this document)	Xcelerator_PropertyLookup.pdf
Installation files	Package/xcp2_propertylookup.jar
Source code	Source

Installing xCP PropertyLookup xCelerator

Prerequisites

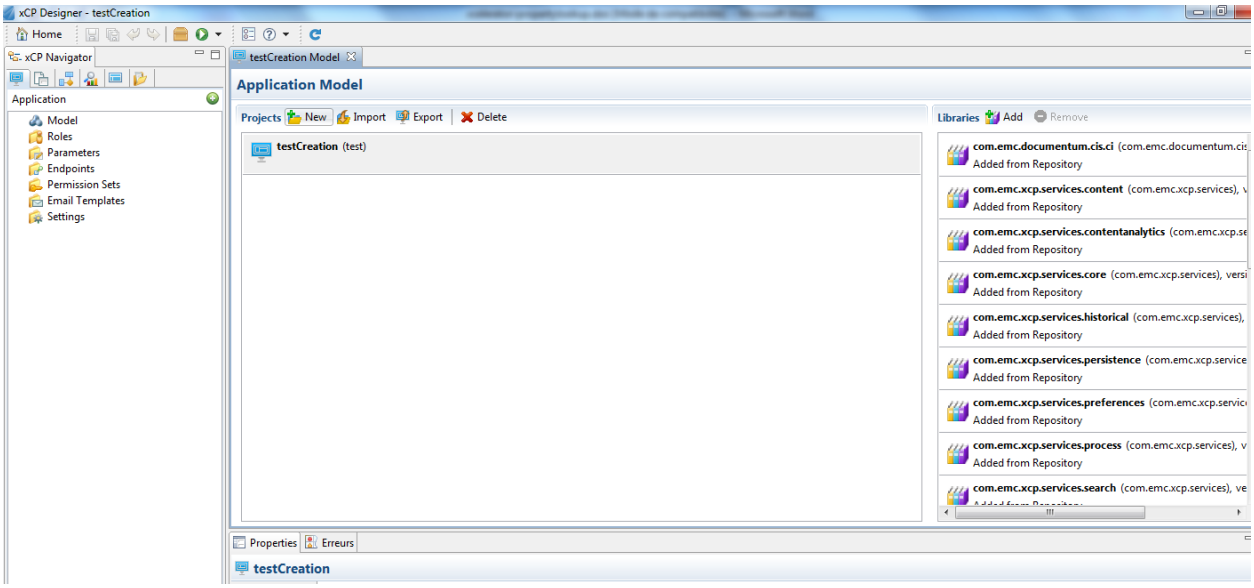
XCP Designer 2.0 must be installed

Install the PropertyLookup xCelerator file

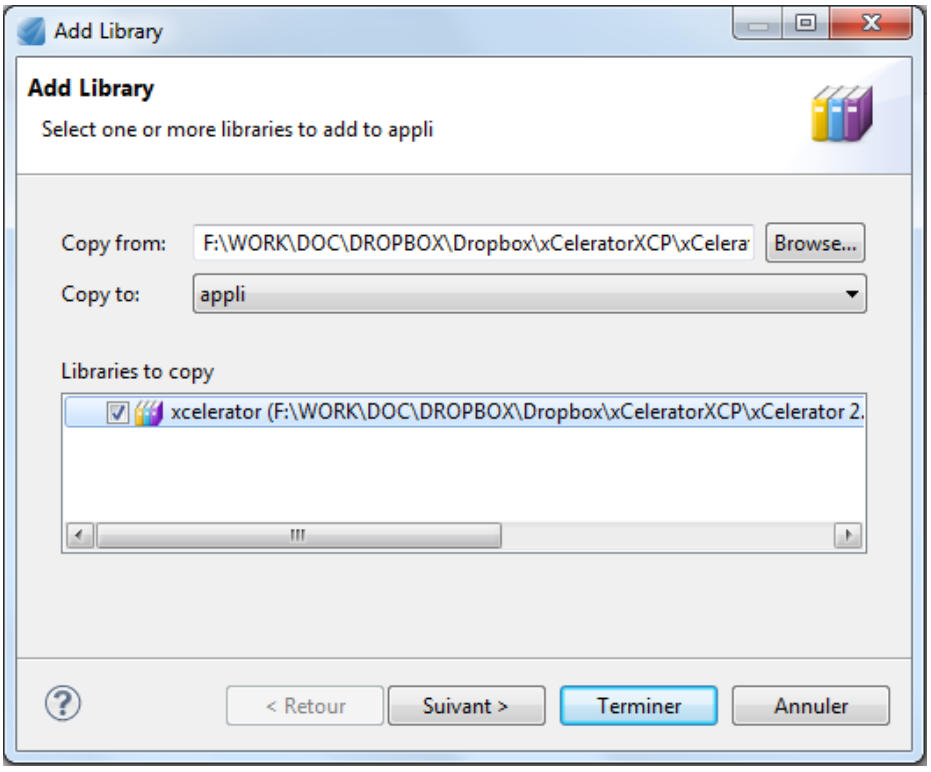
Download the xcp2_propertylookup_xcelerator.zip file which is accessible through the EMC Documentum xCelerator community home page (~~https://community.emc.com/docs/DOC-10586~~) by selecting the PropertyLookup xCelerator link.

Unzip the xcp2_propertylookup_xcelerator.zip file and locate the xcp2_PropertyLookup.jar at following path : *<Install folder>/xcp2_propertylookup_xcelerator/Package/xcp2.propertylookup-1.0.0.jar*

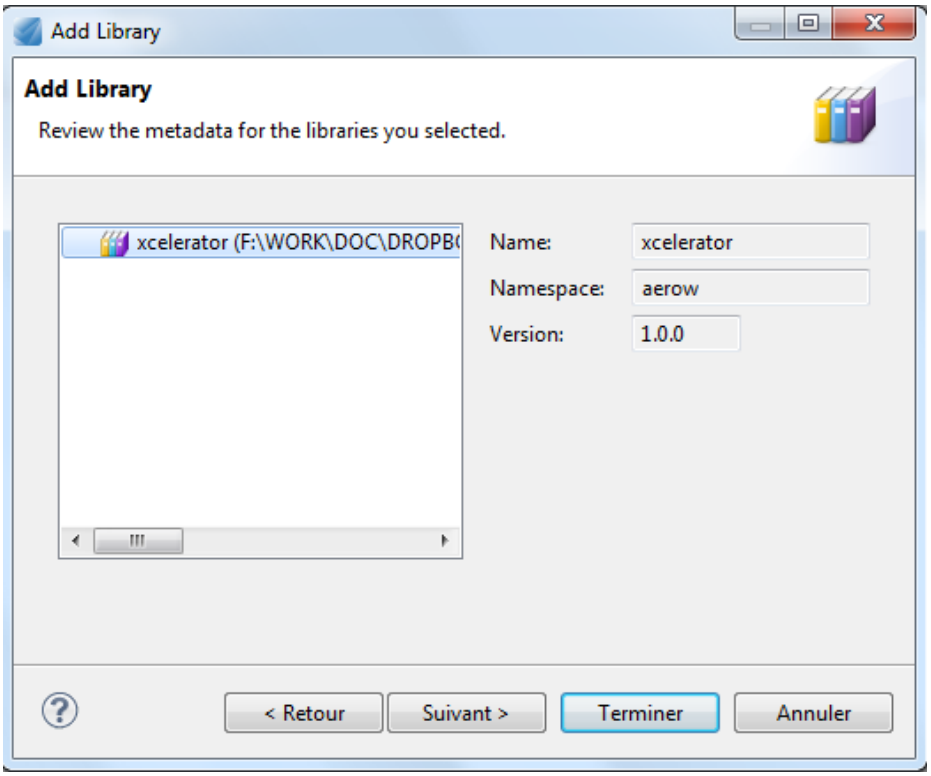
Open your application in xcpDesigner and select “Model” in the “Application” tab:



In the Libraries part, click on the button “Add”. A pop-up will open. Click on the “Browse” button and navigate to the folder *<Install folder>/xcp2_propertylookup_xcelerator/Package* and select xcelerator in “Libraries to copy”. Click on Next button.



Check if the name and the namespace are well filled and click on Finish button.

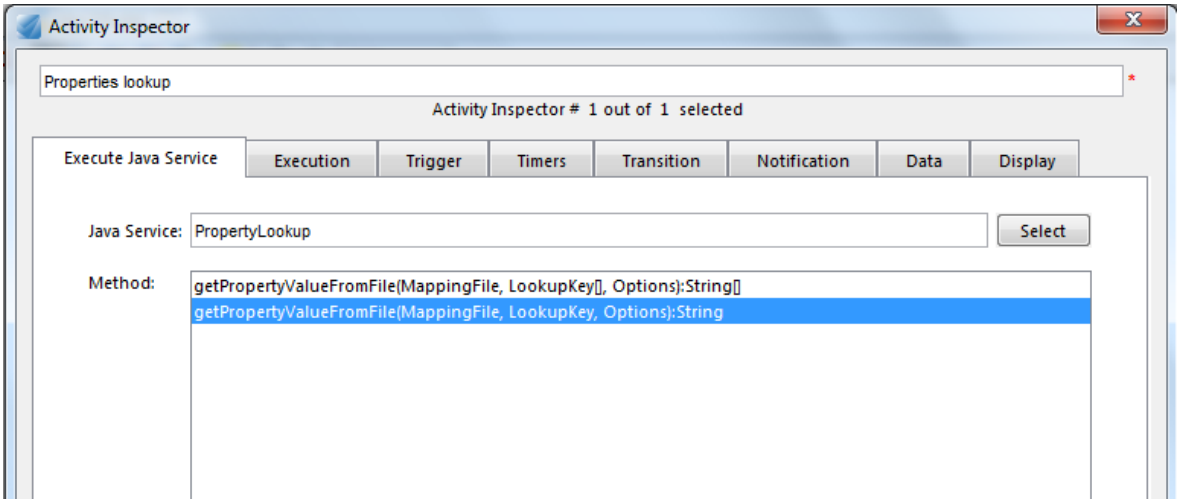


Using the PropertyLookup xCelerator

How to Use PropertyLookup in Your Process

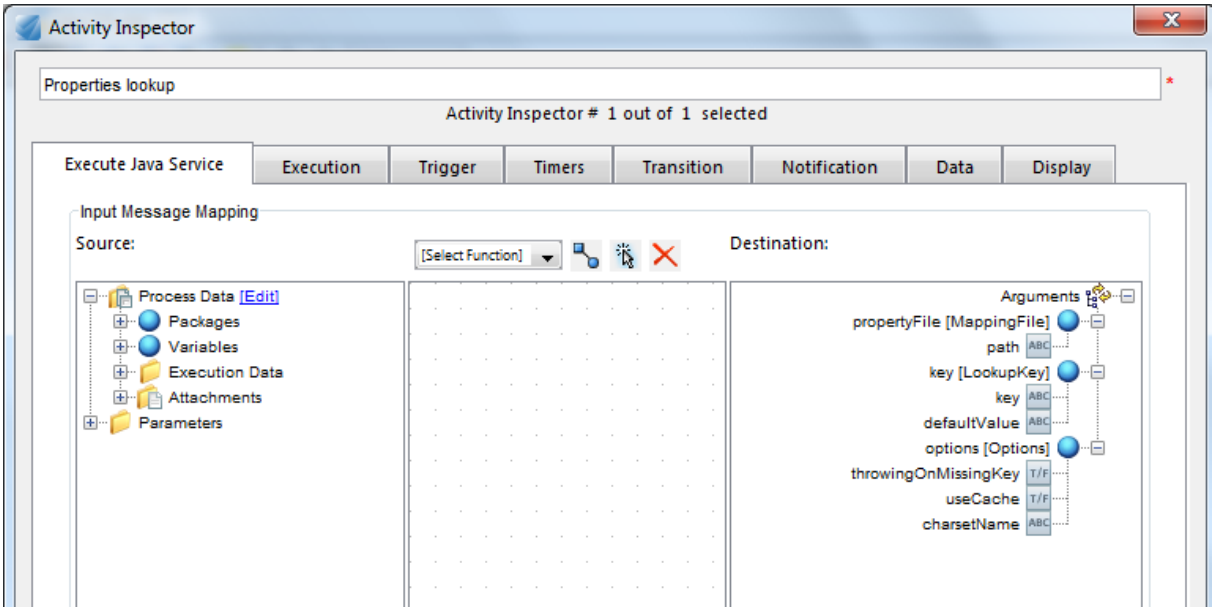
Using the PropertyLookup methods in your process is an easy 4 step process.

Add an Execute Java Service Activity Template to your process. In the Execute Java Service Configuration tab, choose the Property Lookup as the Java Module.

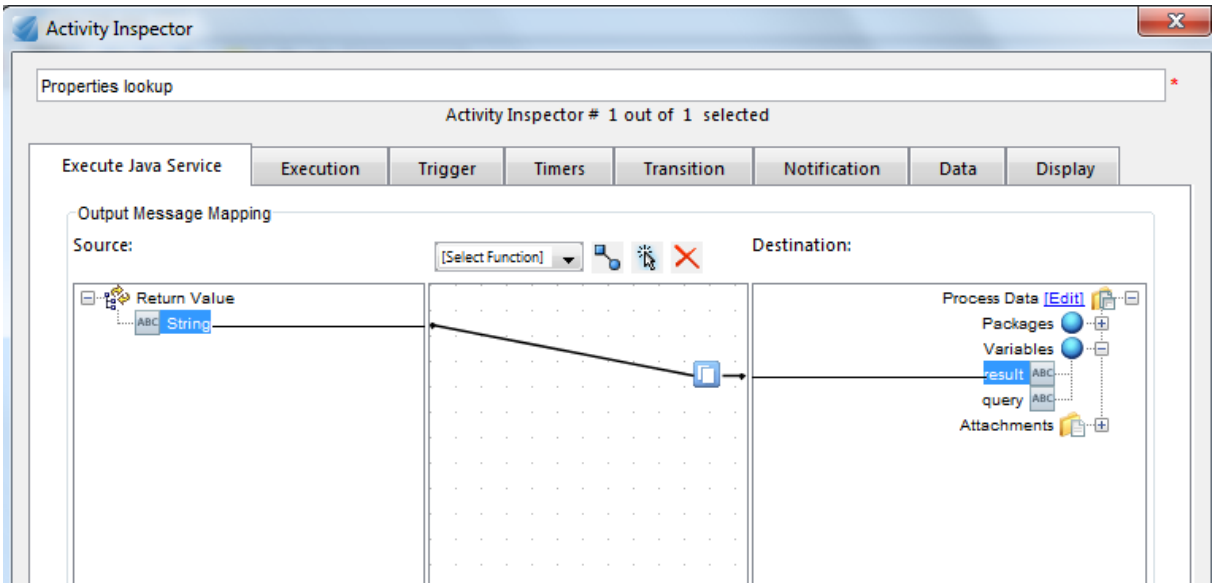


Choose which method you want to invoke (refer to the next chapter for an overview of what methods exists and what argument they take).

Map your process data into the functions.



Map the result into your Process data:



Overview of the provided methods

There are two methods that can be used by the PropertyLookup xCelerator. One that performs a single key-value look up, and one that look up values for multiple keys.

Single key look up

```
String GetPropertyValueFromFile(MappingFile, LookupKey, Options)
```

Use this method when a single key should be looked up by the activity. It will return the value as a string value.

Multiple key look up

```
String[] GetPropertyValueFromFile(MappingFile, LookupKey[], Options)
```

Use this method when a set of keys should be looked up by the activity. It will return a set of string values that can be used in the following activities. The values returned are placed at the corresponding position to their key. That is LookupKey[0] will put the value in String[0], LookupKey[1] in String[1] and so on.

Arguments

Below is a description of the arguments.

MappingFile

The mapping file represents the absolute path within the repository to the file containing the key-value repository to use in the call. You do not have to put the file's extension in the path.

For example, the mapping file value could be `/Temp/samplefile` for the file */Temp/samplefile.properties*.

LookupKey and LookupKey[]

LookupKey is representing the key to find a value from. The LookupKey has two properties, the key itself, and a DefaultValue. The DefaultValue property can be used to return a value that is used if the key was not found. When providing multiple LookupKeys, then each key will have its own default value.

Options

ThrowingOnMissingKey

If this value is set, then the PropertyLookup will throw an exception if the key is missing. This might be used for certain situations. Default is not to throw exceptions but to return the default value provided by the given LookupKey argument.

useCache

Boolean argument describing if caching should be used. Using cache is default value, and it is probably the way to go most of the times. Cache will make the lookup of multiple values from the same MappingFile a little bit shorter since the property file will be stored in the cache when it has been loaded once. Not using cache might be a little bit slower if there are many lookups to the same property file.

charsetName

Name of the character set used in the property file. The default is 'ISO-8859-1'. If omitted this default value will be used.

Property files

The property files that PropertyLookup xCelerator reads are standard java.util.Properties files, that is they are simple text files with the following structure

```
# Line Comments
#
akey=aValue
```

Or

```
# Line Comments
#
akey:aValue
```

Where akey is a string representing a property that will be looked up by the workflow, and aValue is the value that will be returned from this lookup.

For more information about this file format see API documentation for 'java.util.Properties', and especially the 'load(Reader)' method documentation.

Considerations

Caching

The PropertyLookup xCelerator is using cache for property file retrieval. This cache mechanism is looking at the property file modification timestamp to decide if the file should be reloaded or not. There are no restriction of the cache size (except amount of available memory) in this module (may be added in later releases), and this has some consequences.

If there are many different property files accessed, then this xCelerator will eat a lot of memory. So the recommendation would be not to use a lot of different property files. For single property files however this will work nice.

Memory usage

There might be an issue using large property files. This since the PropertyLookup xCelerator will load the whole property file into memory, not just lookup one single property. So the usage of large property files is discouraged.

There are no measurements of how large amount of properties will affect the performance. However, the more properties, the more amount of memory will be used to load these properties. See the API documentation for java.util.Properties for more information.

Escape characters

When working with paths, the property file parser will parse '\' as an escape character. To get around this problem use the '/' as path separator instead. For more information about how the property files are parsed, refer to the API documentation for java.util.Properties