How-To Library

Informatica

Identifying Indirect and Remote Dependencies for an Application Patch in the Developer Tool

Abstract

An applicaton patch can inherit direct, indirect, and remote dependencies. You can identify direct dependencies based on design-time objects, but you must use both the design-time and run-time objects to identify indirect and remote dependencies. This article will present scenarios to demonstrate how you can use the application object instances to identify the dependencies.

Supported Versions

- Big Data Management® 10.2.2
- Big Data Streaming 10.2.2

Table of Contents

Overview

An application patch can inherit indirect and remote dependencies depending on the type of patch that you create and the objects that you select to include in the patch.

If the design-time application and the run-time application have different sets of dependencies, you must observe the dependencies in the both design-time and run-time objects to correctly identify indirect and remote dependencies.

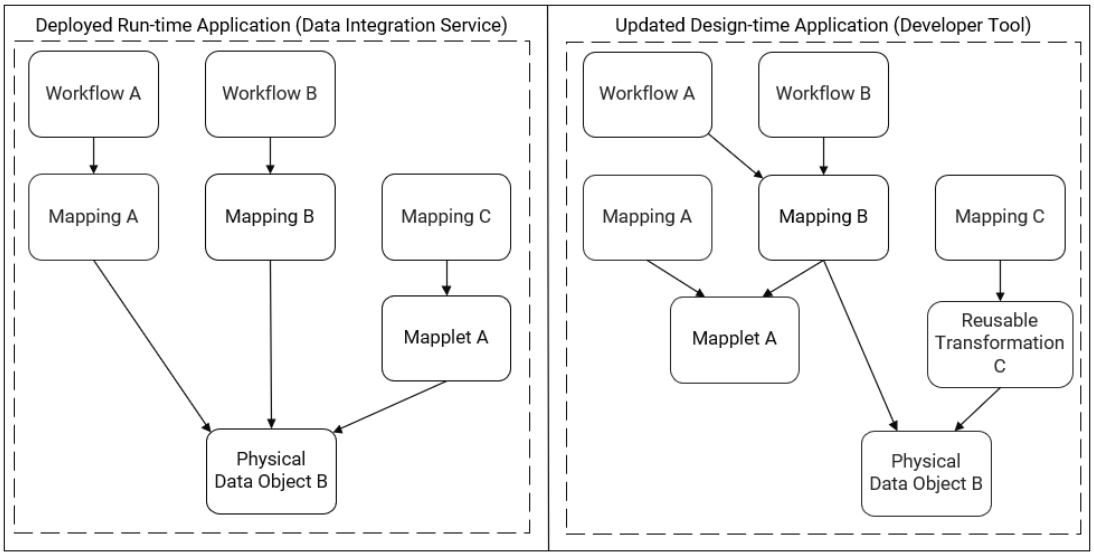
To identify the indirect and remote dependencies, refer to the following steps:

1. Identify the selection and its direct dependencies in the design-time application. This article will refer to this set of objects as the base selection.
2. Match the base selection to its modified objects in the run-time application. The modified objects are run-time instances of the objects in the base selection.
3. Identify the indirect and remote dependencies of the modified objects using the run-time application.
4. Match the indirect and remote dependencies in the run-time application to corresponding objects in the design-time application.

Scenario. Differing Dependencies Across Application Object Instances

The following scenario will use an application with a complex set of dependencies where the dependencies in the deployed run-time application are different from the dependencies in the updated design-time application.

The following image shows the design-time and run-time applications:



The set of dependencies in the deployed run-time application do not match the set of dependencies in the updated design-time application to reflect how design-time and run-time instances of application objects are isolated in data structures. Since the time that the application was deployed, multiple users might have edited the design-time objects and redefined their dependencies without altering the run-time objects.

For example, the deployed application contains a mapping `Mapping B` that uses a data object `Physical Data Object B` and a mapping `Mapping C` that uses a mapplet `Mapplet A`.

Meanwhile, the design-time application has been revised such that the mapping `Mapping B` uses the mapplet `Mapplet A` and the data object `Physical Data Object B`, and the mapping `Mapping C` uses a reusable transformation `Reusable Transformation C` to transform the data in the data object `Physical Data Object B`.

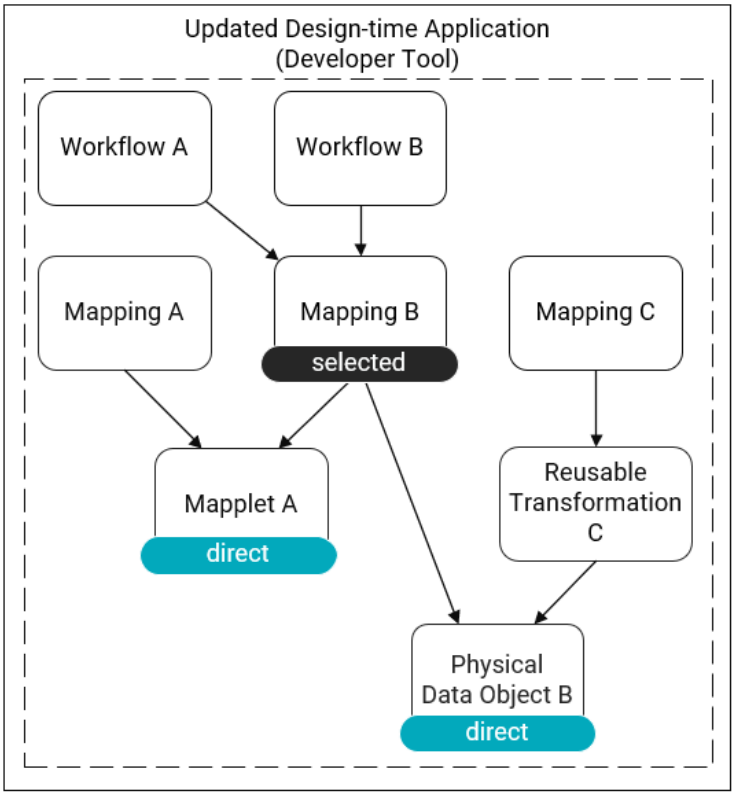
Because the sets of dependencies are different across the design-time and run-time objects, you must use both the design-time and the run-time objects to identify a selected object's indirect and remote dependencies.

Step 1. Identify the Base Selection

To identify the base selection, use the objects that you select to appear in the application patch and the direct dependencies that the patch inherits.

For example, the selected object might be the mapping `Mapping B`. Its direct dependencies are the mapplet `Mapplet A` and the data object `Physical Data Object B`.

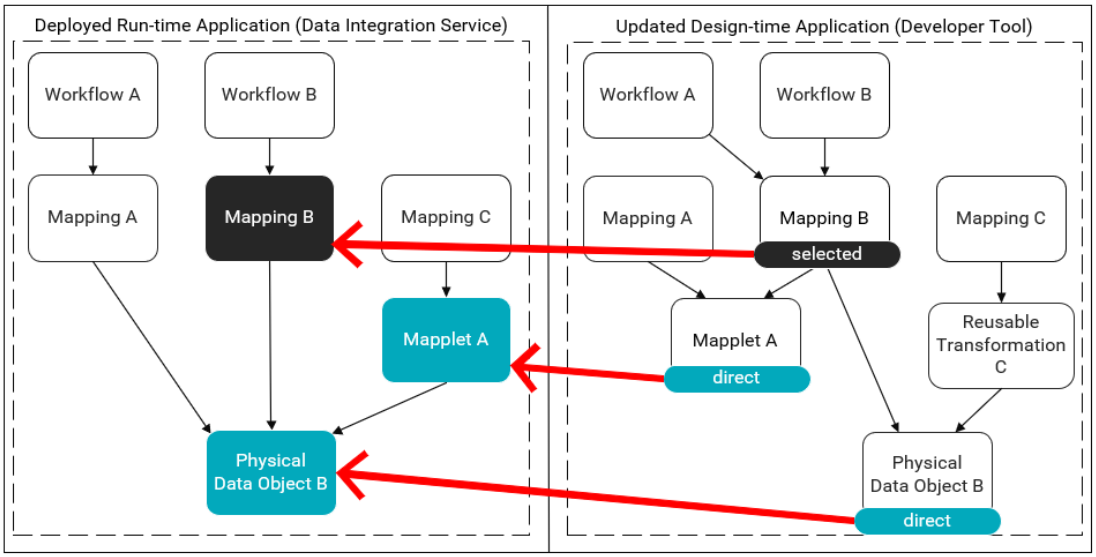
The following image shows the design-time application and labels the mapping and its direct dependencies:



Step 2. Match the Base Selection to Modified Objects

Match the objects in the base selection to corresponding run-time objects that will be modified when you deploy the application patch.

The following image shows design-time objects in the base selection that are matched to the objects that will be modified in the run-time application:

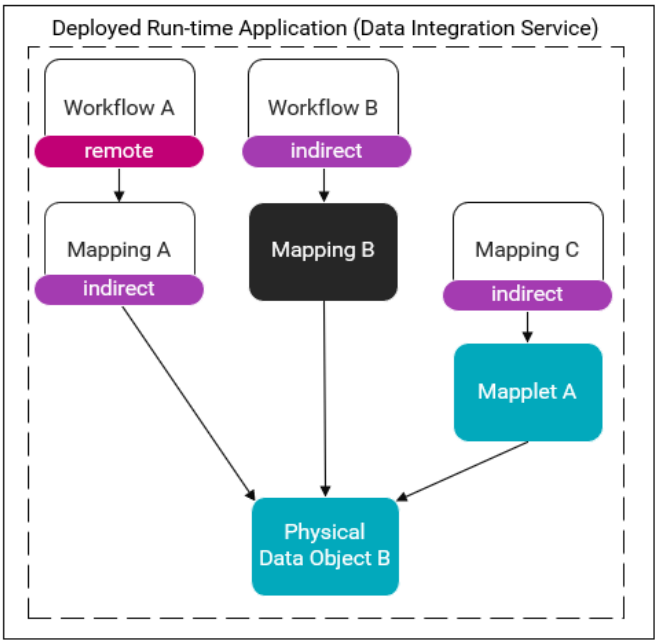


If the application contains multiple objects that share the same name, match the objects by matching the object paths in the design-time and run-time applications.

Step 3. Identify the Dependencies of the Modified Objects

Use the run-time application to identify the indirect and remote dependencies of the modified objects.

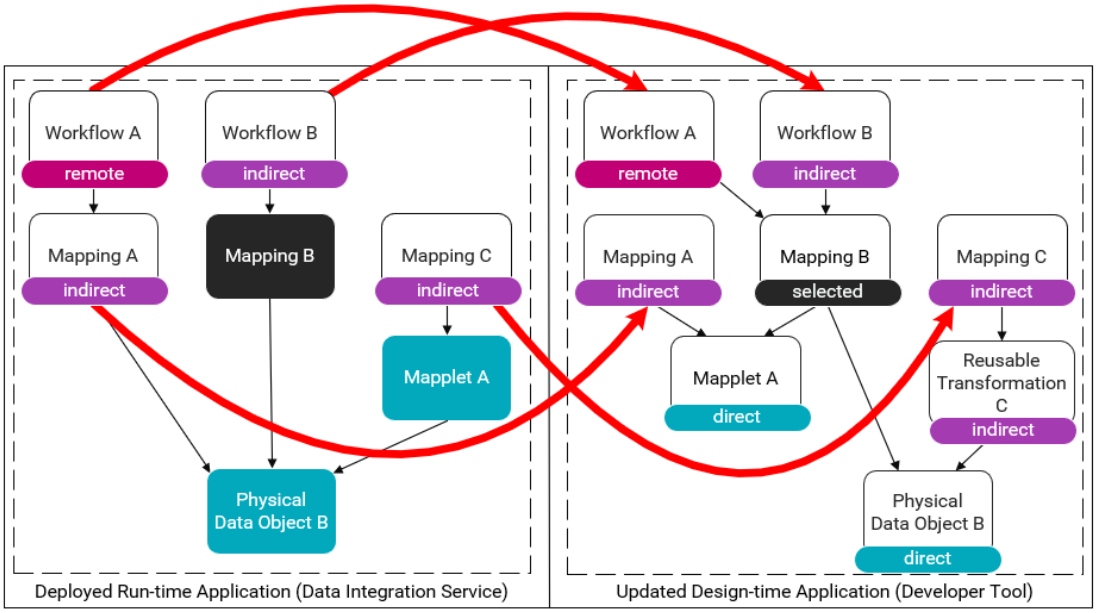
For example, the following image shows the modified objects `Mapping B`, `Mapplet A`, and `Physical Data Object B`, and labels their indirect and remote dependencies in the run-time application:



Step 4. Match the Dependencies to Design-time Objects

After you identify dependencies in the run-time application, match the run-time objects to their corresponding design-time instances. The design-time objects assume the same type of dependency as their run-time instances.

The following image shows how the run-time objects are matched to corresponding design-time objects:



Based on the above image, the mapping `Mapping B` has the following dependencies:

Type of Dependency	Dependencies
Direct	- Mapplet A - Physical Data Object B
Indirect	- Mapping A - Mapping C - Workflow B - Reusable Transformation C
Remote	- Workflow A

Note: The reusable transformation `Reusable Transformation C` becomes an indirect dependency for the mapping `Mapping B`, because it is a direct dependency for the mapping `Mapping C` which is an indirect dependency for the mapping `Mapping B`.

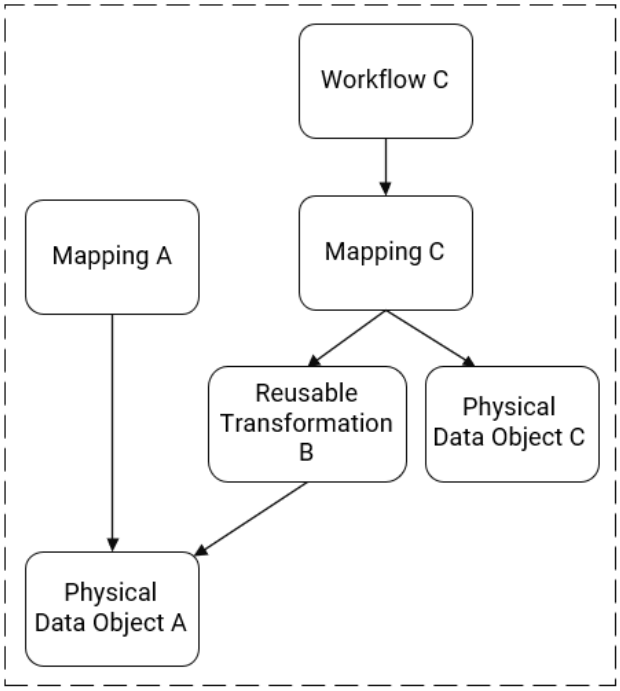
Notice how the indirect and remote dependencies reflect the set of dependencies in the run-time application. For example, the design-time application suggests that the workflow `Workflow A` is an indirect dependency for the mapping `Mapping B` because the workflow uses the mapping in the design-time application. When you examine the run-time application, you can see that the workflow `Workflow A` is actually a remote dependency.

After you have identified the indirect and remote dependencies, you can choose to include the dependencies in the application patch by creating a patch that inherits indirect dependencies or indirect and remote dependencies.

Scenario. Equivalent Dependencies Across Application Object Instances

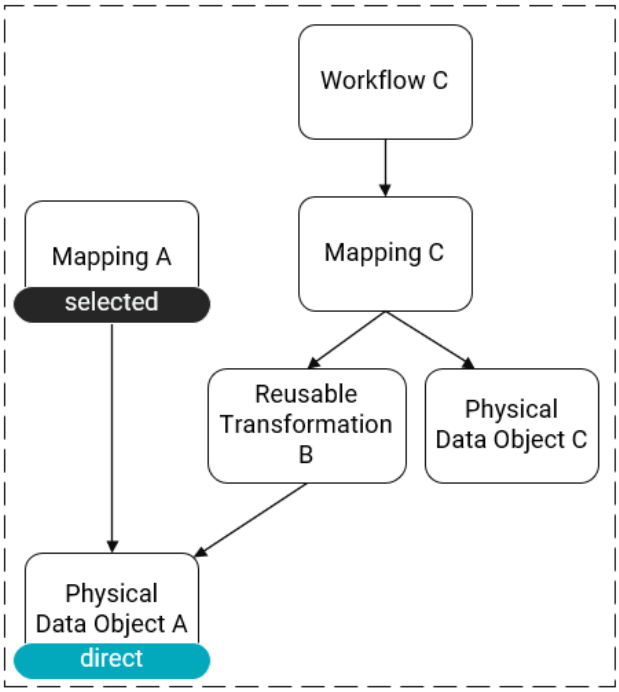
This scenario shows an application that uses the same set of dependencies across the design-time and run-time objects. When the sets of dependencies are the same, you can identify indirect and remote dependencies using only the design-time objects.

The following image shows the set of dependencies across both application object instances:



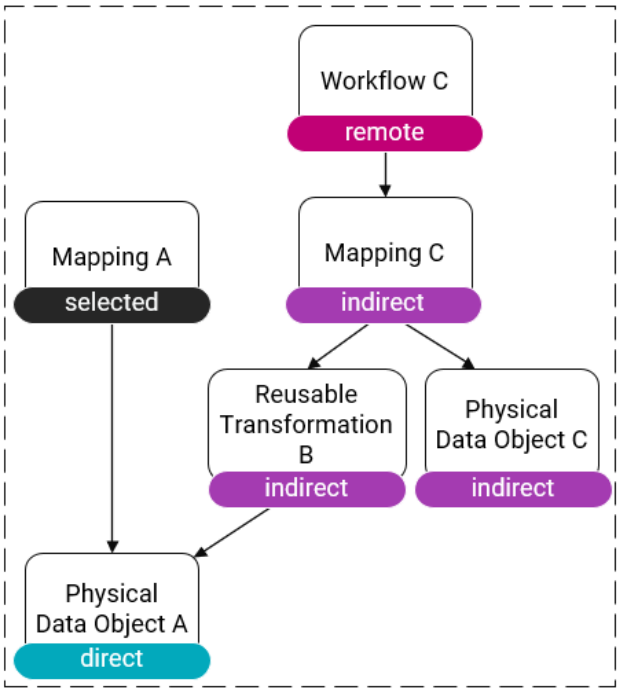
If you select the mapping `Mapping A`, the base selection consists of the mapping and its direct dependency, the data object `Physical Data Object A`.

The following image labels the base selection:



To find indirect and remote dependencies, you can use the design-time objects without consulting the run-time objects.

The following image labels the indirect and remote dependencies:



Based on the above image, the mapping `Mapping A` has the following dependencies:

Type of Dependency	Dependencies
Direct	- Physical Data Object A
Indirect	- Mapping C - Reusable Transformation B - Physical Data Object C
Remote	- Workflow C

Author

Margarita Pelyushenko