Techniques for Deployment from a Versioned PowerCenter Repository
Abstract

Versioned repositories make it easy to deploy workflows and objects from development to production. This article describes versioned objects and how you can use them to deploy workflows. It also provides a sample deployment strategy that shows techniques for how to use versioned objects in deployment.

Supported Versions

- PowerCenter 8.5 - 9.1.0

Table of Contents

Overview ..................................................................... 2
Check-in Comments .............................................................. 3
Label....................................................................... 3
Deployment Group ............................................................... 4
Object Query .................................................................. 4
Global Object Privileges ........................................................... 5
Object History .................................................................. 5
Copy Wizards .................................................................. 5
Deployment Example ............................................................. 6

Overview

If you have the team-based development option, you can enable version control for a repository. PowerCenter version control features allow you to efficiently develop, test, and deploy workflows and objects into production.

A versioned repository can store multiple versions of the same object. Each time you check in an object, the PowerCenter Repository Service adds a new version of the object to the repository.

You can store multiple versions of the following objects in a versioned repository:

- Sources and targets
- Transformations
- Mappings and mapplets
- Sessions and session configurations
- Tasks, workflows, and worklets
- User-defined functions
- Schedulers
- Cubes and dimensions

If you have multiple versions of objects in a development repository, you can select the version to deploy into production and use global objects to make the deployment easier. You can deploy objects to a different folder in the same repository or to a different repository.

A versioned repository provides many features that you can use to make deployment easier. Consider including these features when you develop your deployment strategy:
Check-in comments

Use comments to provide specific information for each version of an object that you check in.

Label

Use a label to easily identify an object. You can also use the same label for a group of objects to identify related objects.

Deployment group

Use a deployment group to identify a group of objects to be deployed at the same time.

Object query

Use an object query to search for repository objects that have common attributes.

Object privileges

Set the privileges and permissions on the global objects that you use in deployment to minimize unauthorized access.

Object history

View the object history to keep track of changes made to each version of an object.

Copy wizards

Use the Copy Folder Wizard or the Copy Deployment Group Wizard to facilitate the deployment of objects from one repository to another.

Check-in Comments

When you check in a version of an object, you can add comments to indicate the changes made to the object. A comment is associated with the specific version of the object that you check in.

You can use comments to provide significant information for each version of the object in the repository. You can use the same comment for related objects that you check in at the same time.

You can turn on the option to require comments at every check-in. For comments to be useful, they must include detailed information about the changes being checked in. The more useful information included in the comments, the more beneficial it is to the deployment process.

Label

A label is a global object that you can associate with a version of an object or group of objects.

Labels provide a way to tag a version of an object for identification. You can use the label to identify the version of an object or a group of objects that is included in a release or objects that are ready for deployment. For example, you might apply a label to sources, targets, mappings, and sessions associated with a workflow to identify all the objects that must be deployed with the workflow to maintain object dependency.

A label is associated with a specific version of an object. For example, if there are ten versions of a mapping checked in and you apply a label to version nine, then only version nine has that label. Older versions of the object are not associated with the label and later versions of the object do not inherit that label.

You can include labels as part of a query condition to easily identify a set of objects. You can then use the object query to build a deployment group. You can also export or view the history of a group of objects with the same label.

Lock a label to prevent users from modifying the list of objects associated with it or from associating the label with the wrong object version. Locking a label to control the version of objects associated with it can minimize errors in deployment that could cause failures in production.
When you create labels, it is useful to define a labeling convention. Consistent labels make it easier to see or select labels because related labels display in sequence on the Label Browser. They also make it easier to define query conditions that include the labels.

**Deployment Group**

A deployment group is a global object that contains references to versions of objects from one or more folders.

A deployment group provides a means to deploy a set of objects from different folders in a single operation. You can use a deployment group to deploy objects from one or more folders in a versioned repository to another versioned repository.

A deployment group can be one of the following types:

- Static. Contains a list of specific versions of objects. The list can include dependent objects.
- Dynamic. Contains a variable list of objects. An object query associated with the deployment group determines the list of versioned objects that is included in the group.

If you know the objects you want to deploy and the list of objects is small, use a static deployment group.

If you do not know the precise list of objects or if the list of objects is long, use a dynamic deployment group. The object query associated with the deployment group runs at the time of deployment and creates the list of objects to be deployed.

To protect the integrity of repository objects, you cannot copy a deployment group when objects included in the deployment group are checked out or locked in the target repository. Before you copy a deployment group, search for checked out objects in the target repository and verify that no deployment target objects are checked out. You can freeze folders in the target repository to ensure that no object is checked out when you copy a deployment group. When you freeze a folder, you prevent other users from checking out objects in the folder, but you can still use the copy wizard to copy and check in deployment group objects.

After you copy a deployment group, verify that the objects and dependent objects are valid in the target repository. If the deployed objects are not valid, you can roll back a deployment group to purge the deployed versions of the objects from the target repository.

**Note:** Effective PowerCenter 8.6.1, you can use deployment groups to copy objects in versioned and non-versioned repositories. For versioned repositories, the copy operation creates new versions of existing objects in the target folders. For non-versioned repositories, if the objects in the deployment group exist in the target repository, the copy operation deletes existing objects and creates new objects in the target repository.

**Object Query**

You can use an object query to search for repository objects that meet specified conditions.

Use an object query to identify objects that share the same attributes. For example, you can create a query that identifies all mappings that are dependent on a specific source definition or all objects that have a specific version in the label. You can also include check-in metadata such as user name and date in queries to track versioned objects during development.

Query conditions can be combined and nested using Boolean AND and OR operators.

You can associate a query with a deployment group to create a dynamic deployment group. You can also view the history of an object returned by a query, view the dependencies of the object, or compare the object with another version of the object or with another object.

Use the Query Browser to create and manage object queries. Because queries can be used for different purposes, define a naming convention to make it easier to determine what a query is used for. For example, you can use the prefix “Deploy_” to clearly identify queries that are used in a dynamic deployment group.
Global Object Privileges

You can set privileges on global objects to control access to objects you use in deployment.

The privileges and permissions set for PowerCenter global objects determine the tasks users can perform on the following objects:

- Deployment groups
- Labels
- Queries

Setting the privileges on the objects allows the administrator to limit user access to the objects. You can create global objects that can be used only by specific users and groups.

For example, you can set the object privileges so that only users who work on a specific project have read, write, and execute permissions for labels, queries, and deployment groups used in that project. Or you can set permissions for a query so only certain users can run the query. Setting permissions on a query does not prevent the query from appearing in the query browser, but it prevents unauthorized users from accessing the query.

Object History

The history of an object is a record of all of the versions of an object stored in the repository, going back to the initial version.

The information stored for each version of an object provides an audit trail for the object over time. It includes the changes to the object, the date and time of the changes, and the comments added during check-in. It also includes the labels applied to each version. When an object version is deleted from the repository, the object history retains a record of the object version.

If you notice a change in workflow behavior that is not intentional and must be corrected, the audit trail allows you to trace changes back through previous deployments and determine the exact version that caused the change. Check-in comments with detailed information about each change can be a valuable aid in determining the version where the change was added. Object history can also be used to show adherence to development processes and policies during a business audit.

Copy Wizards

To deploy objects into production, you can copy and replace an entire folder or copy the objects in a deployment group.

You can replace a folder in versioned or non-versioned repositories. For example, you build a workflow in a folder called Sales in the development repository. When the objects in the folder are ready for production, you copy the Sales folder into the production repository. After a week in production, you want to make minor changes. You edit the objects in the Sales folder in the development repository and test the changes. When the objects are ready for production, you can copy and replace the Sales folder in the production repository with the updated folder in the development repository. When you replace a folder, you update the target repository without creating multiple copies of the folder.

You can also copy the objects in a deployment group. When you use a deployment group, you can copy objects in a single copy operation from multiple folders in the source repository into multiple folders in the target repository. You can specify individual objects to copy, rather than the entire contents of a folder. Make sure that you include all dependent objects. If you copy objects without their dependent objects and the dependent objects do not exist in the target repository, the copy operation fails.

You can use different copy operations together. You might use copy folder and copy deployment group together or at different stages of the deployment process. For example, when you complete the initial development in a new folder and you are ready to deploy the objects into production, you can copy the folder and all of its objects into the production repository. As development continues, you make changes to a session in the folder. Because you do not need to copy all of the objects in the folder to production, you can create a deployment group and add only the updated session to the deployment group. When you copy the deployment group, you create a new version of the session in the target folder.

The PowerCenter Repository Service provides wizards to copy and replace folders and to copy deployment groups. Use the Copy Folder Wizard to copy or replace a folder and the Copy Deployment Group Wizard to copy a deployment group.
Deployment Example

You can use the PowerCenter version control features to deploy workflows and objects into production more easily and efficiently.

For example, your environment includes a development repository and a production repository. You create and test workflows and transformation objects in the development repository before you deploy them to the production repository. After the initial deployment, you update a workflow in the development repository and deploy the new version to the production repository.

To use the versioned objects to deploy workflows and objects in the scenario described, complete the following tasks:

**Step 1. Use Versioned Repositories for Development and Production**
Create, test, and check in workflows and objects in the development repository. When you check in an object, include comments that describe the object and explains the changes to the object.

**Step 2. Create a Label for the Version**
Specify a label name and add comments and tags relevant to the version that could be used in searches and queries. Apply the label to all objects that you want to deploy as part of the same version. Set permissions on the label to limit access only to specific users and to prevent other users from editing or applying the label to unrelated objects.

**Step 3. Create an Object Query for a Dynamic Deployment Group (Optional)**
If you have a long list of objects to deploy, you can associate an object query with a deployment group and use the query to determine the objects to be added to the deployment group. Create an object query that returns the list of all the objects to be deployed.

**Step 4. Use the Copy Wizards to Deploy**
If you create all objects in one folder, use the Copy Folder Wizard to copy the entire folder from the development repository to the production repository.

If you create the objects in multiple folders, create a deployment group and include all the objects to be deployed together. If the list of objects to be deployed is small and static, you can individually add the objects to the deployment group. Ensure that all dependent objects are included in the deployment group. If the list of objects is long or the objects can be identified based on an attribute, use an object query to add the objects to the deployment group. Then use the Copy Deployment Group Wizard to copy the objects from the development repository to the production repository.

**Step 5. Update a Workflow in the Development Repository**
When you need to update a workflow, make the changes to the workflow in the development repository.

If you encounter a problem with a workflow, view the object history to trace the changes made to the objects in the workflow.

**Step 6. Use the Copy Wizards to Redeploy**
If you copied an entire folder during the initial deployment, create a deployment group and add only the updated objects. Then use the Copy Deployment Group Wizard to deploy the updated objects to the production repository. The wizard creates new versions of the updated objects in the production repository.

If you used a deployment group during the initial deployment, you can create another deployment group for the new release and add only the updated objects. Then use the Copy Deployment Group Wizard to deploy the updated objects to the production repository. You can also use the original deployment group to deploy the objects to the production repository. If you use the original deployment group, the Copy Deployment Group Wizard creates a new version in the production repository for all objects in the deployment group.

**Step 7. Verify the Updated Objects in the Target Repository**
After you copy a deployment group, verify that the deployed objects and dependent objects are valid in the production repository. Create an object query to search for all updated objects in the target repository and verify that they are valid.
You can use the Validate Target Repository option in the copy wizard to verify that the deployed objects are valid. However, if you use the Validate Target Repository option, the wizard validates the whole target repository, not just the updated objects. The validation can take some time.

Author

Marissa Johnston
Principal Technical Writer

Acknowledgements

Thanks to Samir Pradhan and Kamakshi Ram for their contributions to this article.