25

Resource Versions

Oracle XML DB Repository resources can be versioned. A record is kept of all changes to a resource that is under version control.

> **Note:**
>
> The Oracle XML DB Repository is deprecated with Oracle Database 23ai.

- Overview of Oracle XML DB Repository Resource Versioning
  You can create and manage different versions of a repository resource. A record is kept of all changes to a resource that is under version control. When you update a version-controlled resource, the pre-update contents are stored as a separate resource version – a snapshot for the historical record.
- Overview of PL/SQL Package DBMS_XDB_VERSION
  You use PL/SQL package `DBMS_XDB_VERSION` to work with Oracle XML DB Repository resource versions.
- Resource Versions and Resource IDs
  A resource object ID, or **resource ID**, is a unique, constant, system-generated identifier for a resource. Each resource has a resource ID. This includes *version resources*, which are system-generated resources that do not have any path names. A resource ID is sometimes called a **RESID**.
- Resource Versions and ACLs
  A version resource is immutable. It is a snapshot of resource content and metadata, plus a resource ID, and both snapshot and ID are static. Likewise, the access control list (ACL) of a version resource cannot be changed.
- Resource Versioning Examples
  Examples here create a version-controlled resource; retrieve the content of a resource using its resource ID; check out a version-controlled resource (for all users); update the content of a resource; check in a resource; retrieve the content and metadata of different versions of a resource; and cancel a resource check-out.

Overview of Oracle XML DB Repository Resource Versioning

You can create and manage different versions of a repository resource. A record is kept of all changes to a resource that is under version control. When you update a version-controlled resource, the pre-update contents are stored as a separate resource version – a snapshot for the historical record.

Versioning features include the following:

- Version control for a resource.

  You can turn version control on or off for an Oracle XML DB Repository resource.
- Updating a version-controlled resource.

When Oracle XML DB updates a version-controlled resource, it creates a new version of the resource. This new version is not deleted from the database when you delete the version-controlled resource.

- Accessing a version-controlled resource.

  You can access a version-controlled resource the same way you access any other resource.

- Accessing a resource version.

  To access a particular version of a resource, you use the resource ID of that version. The resource ID can be obtained from the resource version history or from the version-controlled resource itself. See Resource Versions and Resource IDs.

Table 25-1 lists some terms used in this chapter.

Table 25-1 Oracle XML DB Versioning Terms

Term	Description
Versionable resource	A resource that can be put under version control. All Oracle XML DB resources except folders and ACLs are versionable.
Version-controlled resource	A resource that is under version control.
Version resource	A particular version of a version-controlled resource. A version resource is itself a resource. It is system-generated, and it has no associated path name. It is read-only (it cannot be updated or deleted).
`checkOut`, `checkIn`, `unCheckOut`	Operations for managing version-controlled resources. You must use `checkOut` before you can modify a version-controlled resource. Use `checkIn` to make your changes permanent. Use `unCheckOut` to cancel your changes. (Use `COMMIT` after each of these operations.)

Note:

Oracle XML DB supports version control only for Oracle XML DB resources. It does *not* support version control for user-defined tables or data in Oracle Database.

Oracle does *not* guarantee preservation of the resource ID of a version across check-in and check-out. Everything except the resource ID of the latest version is preserved.

Oracle XML DB supports versioning of XML resources that are not XML schema-based. It also supports versioning of XML schema-based resources and resources that contain XML schema-based metadata, but only if the underlying tables have *no* associated triggers or constraints.

If hierarchy is enabled for a table, then the table has a trigger. This includes tables that are created as part of XML schema registration, for which the default behavior is to enable hierarchy.

Be aware also that if you query one of the tables underlying a resource, the query can return data from multiple versions of the resource. This is because the data for the different resource versions is stored in the same underlying table, using different rows.

Overview of PL/SQL Package DBMS_XDB_VERSION

You use PL/SQL package `DBMS_XDB_VERSION` to work with Oracle XML DB Repository resource versions.

Table 25-2 summarizes the main `DBMS_XDB_VERSION` subprograms.

Table 25-2 PL/SQL Functions and Procedures in Package DBMS_XDB_VERSION

Function or Procedure	Description
`makeVersioned(pathname VARCHAR2) RETURN DBMS_XDB_VERSION.RESID_TYPE;`	Turn a resource with the given path name into a version controlled resource. If two or more path names refer to the same resource, then the resource is copied, and argument path name is bound with the copy. The new resource is put under version control. All other path names continue to refer to the original resource. The argument is the path name of the resource to be put under version control. Returns the resource ID of the first version resource of the version-controlled resource. This is not an auto-commit SQL operation. An error is raised of you call `makeVersioned` for a folder, version resource, or ACL, or if the target resource does not exist. Note: No error or warning is raised if you call `makeVersioned` for a version-controlled resource.
`checkOut(pathname VARCHAR2);`	Check out a version-controlled resource. You cannot update or delete a version-controlled resource until you check it out. Check-out is for all users: any user can modify a resource that has been checked out. The argument is the path name of the version-controlled resource to be checked out. This is not an auto-commit SQL operation. If two users check out the same version-controlled resource at the same time, then one user must roll back. As a precaution, commit after checking out and before updating a resource. An error is raised if the target resource is not under version control, does not exist, or is already checked out.
`checkIn (pathname VARCHAR2) RETURN DBMS_XDB_VERSION.RESID_TYPE;`	Check in a version-controlled resource that has been checked out. `pathname` - Path name of the checked-out resource. Returns the resource id of the newly created version. This is not an auto-commit SQL operation. You need not use the same path name that was used for check-out. However, the check-in path name and the check-out path name must reference the same resource, or else results are unpredictable. If the resource has been renamed, then the new name must be used when checking it in. An error is raised if the path name refers to no resource.
`unCheckOut(pathname VARCHAR2) RETURN DBMS_XDB_VERSION.RESID_TYPE;`	Check in a checked-out resource. The argument is the path name of the checked-out resource. Returns the resource id of the version before the resource was checked out. This is not an auto-commit SQL operation. You need not use the same path name that was used for check-out. However, the `unCheckOut` path name and the check-out path name must reference the same resource, or else results are unpredictable. If the resource has been renamed, then the new name must be used for `unCheckOut`. An error is raised if the path name refers to no resource.
`getPredecessors(pathname VARCHAR2) RETURN RESID_LIST_TYPE;` `getPredsByRESID(resid DBMS_XDB_VERSION.RESID_TYPE) RETURN RESID_LIST_TYPE;`	Given a path name that references a version resource or a version-controlled resource, return the predecessors of the resource. Retrieving predecessors by resource ID, using function `getPredsByRESID` is more efficient than by path name, using function `getPredecessors`. The list of predecessors returned has only one element (the parent): Oracle XML DB does not support version branching.

Table 25-2 (Cont.) PL/SQL Functions and Procedures in Package DBMS_XDB_VERSION

Function or Procedure	Description
`getSuccessors(pathname VARCHAR2) RETURN RESID_LIST_TYPE;`	Given a version resource or a version-controlled resource, return the successors of the resource.
`getSuccsByRESID(resid DBMS_XDB_VERSION.RESID_TYPE) RETURN RESID_LIST_TYPE;`	Retrieving successors by resource ID, using function `getSuccsByRESID` is more efficient than by path name, using function `getSuccessors`. The list of successors returned has only one element (the parent): Oracle XML DB does not support version branching.
`getResourceByRESID(resid DBMS_XDB_VERSION.RESID_TYPE) RETURN XMLType;`	Given a resource ID, return the resource as an `XMLType` instance.

Resource Versions and Resource IDs

A resource object ID, or **resource ID**, is a unique, constant, system-generated identifier for a resource. Each resource has a resource ID. This includes *version resources*, which are system-generated resources that do not have any path names. A resource ID is sometimes called a **RESID**.

You use PL/SQL package `DBMS_XDB_VERSION` to put a resource under version-control and manage different versions of it. Some of the `DBMS_XDB_VERSION` routines accept the path name of a version-controlled resource as argument and return the resource ID of the relevant version resource.

For example, you use function `DBMS_XDB_VERSION.makeVersioned` to put a resource under version control, that is, to turn it into a version-controlled resource. It accepts as argument a repository path to the resource.

You need not use the same path name for a given version-controlled resource when you perform various versioning operations on it, but the path names you use must all refer to the same resource.

Whenever a path name is passed as an argument representing a version-controlled resource, it is the latest (that is, the current) version of the resource that is used. A *path name always stands for the latest version.* The only way you can refer to a version other than the current version is to use its resource ID.

The resource ID of a given version is constant. Remember that a version is itself a resource, and the resource ID of a resource never changes.

Each time you check in a version-controlled resource, Oracle XML DB creates a new version resource. A **version resource** is a snapshot of a resource (its content and metadata) together with a resource ID. The collection of version resources for a given version-controlled resource constitutes a historical sequence of previous versions, the **version series** or history of the resource.

When you check in a version-controlled resource that has resource ID ***R***, Oracle XML DB creates a new resource ID, ***P***, which refers to a snapshot of the resource (both content and metadata), as it was before it was last checked out. The snapshot was made before check-out, but the associated version resource (and its resource ID ***P***) are created at check-in time. Together, the new resource ID ***P*** and the snapshot it refers to thus represent the *previous*, not the current, version of the resource. Resource ID ***R*** continues to refer to the current version.

Put another way, when you check in a version-controlled resource, a version resource is created that represents the previous state of the version-controlled resource. Like any new resource, this new version resource is allocated a new resource ID (***P***).

You can think about making a version resource (check-in) the way you think about making a backup copy of a file: Just as you give a new name to the backup file, so the previous-version snapshot of a resource is given a new resource ID. The current resource retains the original resource ID, just as your working file keeps its original name.

What this means is that when you check in a resource, in order to "create a new version", what's really new is the version resource (resource ID ***P*** and the snapshot it references) that represents the *old* (previous) version. The newest, or latest, version of the resource (***R***) is really just the current version. Remember: new version resource = old (previous) version of the resource content and metadata.

Resource ID ***R*** refers to the *current* version of the version-controlled resource throughout its lifetime, from the moment it was put under version control until it is deleted. You can always access the latest version of a resource using its original resource ID.

When you need to refer to a previous version of a resource, you must use its resource ID to reference it. You cannot use a path name. You can use function `DBMS_XDB_VERSION.getPredsByRESID` to obtain the resource ID of the previous version of a given resource.

> **Note:**
>
> If you *delete* a resource, then any subsequent reference to it, whether by resource ID or path name, raises an error (typically `ORA-31001: Invalid resource handle or path name`). You *cannot* access any version of a version-controlled resource that has been deleted.

Resource Versions and ACLs

A version resource is immutable. It is a snapshot of resource content and metadata, plus a resource ID, and both snapshot and ID are static. Likewise, the access control list (ACL) of a version resource cannot be changed.

You can modify the ACL of a version-controlled resource that you have checked out. When you check it in, the modified ACL continues to be associated with the current (latest) version of the resource, and the previous version, that is, the newly created version resource, is associated with the ACL before it was modified. That is, the previous version is associated with the previous ACL, and the current version is associated with the updated ACL.

What is important to keep in mind is this:

- Different versions of a resource can have different ACLs associated with them.
- You can modify the ACL associated with the current version after you check out the resource.
- Check-in associates the ACL as it was before check-out with the newly created version resource, that is, with the previous version of the resource.
- The ACL associated with a given version remains the same.

Resource Versioning Examples

Examples here create a version-controlled resource; retrieve the content of a resource using its resource ID; check out a version-controlled resource (for all users); update the content of a resource; check in a resource; retrieve the content and metadata of different versions of a resource; and cancel a resource check-out.

- Putting a resource under version control – Example 25-2
- Retrieving the content of the resource using its resource ID – Example 25-3
- Checking out a version-controlled resource (for all users) – Example 25-4
- Updating the content of a resource – Example 25-5
- Checking in a resource – Example 25-6
- Retrieving the content and metadata of different versions of a resource – Example 25-7, Example 25-8, Example 25-9
- Canceling a resource check-out – Example 25-10

Example 25-3 creates an Oracle XML DB Repository resource at repository path `/public/t1.txt`. The resource has as content the text `Mary had a little lamb`. The example uses SQL*Plus command `VARIABLE` to declare bind variables `targetPath`, `current_RESID`, and `previous_RESID`, which are used in other examples in this section.

The new resource is *not* version-controlled. Example 25-2 uses PL/SQL function `DBMS_XDB_VERSION.makeVersioned` to put it under version control. This function returns the resource ID of the first version resource for the version-controlled resource. The function does not auto-commit. You must explicitly use `COMMIT`.

Example 25-2 also copies the resource ID of the new version resource to bind variable `current_RESID`. Example 25-3 shows how to use PL/SQL constructor `XDBUritype` together with PL/SQL function `createOIDPath` to retrieve the resource content by referencing the resource ID.

Example 25-4 checks out the version-controlled resource (and commits), so that it can be modified. Any user can modify a resource that has been checked out.

Example 25-5 updates the content of the checked-out resource. Before the (LOB) content can be updated, you must lock the resource. The example uses a dummy update of the resource display name (a scalar attribute) to do this.

Example 25-5 retrieves the LOB content using the LOB locator, which is element `/ns:Resource/ns:XMLLob`. It empties the existing content and adds new content using PL/SQL procedures `trim` and `append` in package `DBMS_LOB`. It commits the content change.

> **See Also:**
>
> *Oracle Database SecureFiles and Large Objects Developer's Guide* for information about updating a LOB

At this point, the content has been modified, but this change has not been recorded in the version series. Example 25-6 checks in the resource and commits the check-in.

PL/SQL function `checkIn` returns the resource ID of the current version, which is the same as `current_RESID`. Example 25-6 passes this value to PL/SQL function `getPredsByRESID`. This function returns the list of resource IDs for the (immediate) predecessors of its argument resource.[1] Example 25-6 assigns the first (and only) element of this list to bind variable `previous_RESID`.

At this point, the value of `current_RESID` is the resource ID of the current version, and the value of `previous_RESID` is the resource ID of the previous version.

You can retrieve the content or metadata of a resource using any of the following methods:

- PL/SQL constructor `XDBURIType`, together with PL/SQL function `DBMS_XDB_REPOS.createOIDPath` – Retrieve content. See Example 25-3 and Example 25-7.
- PL/SQL function `DBMS_XDB_VERSION.getContentsCLOBByRESID` – Retrieve content. See Example 25-8.
- PL/SQL function `DBMS_XDB_VERSION.getResourceByRESID` – Retrieve metadata. See Example 25-9.

You can use constructor `XDBURIType` with function `createOIDPath` to access resource content using protocols. For example, you could have Oracle XML DB serve up various versions of a graphic image file resource for a Web page, setting the `HREF` for the HTML `IMAGE` tag to a value returned by `createOIDPath`.

Example 25-7 through Example 25-9 use these different methods to retrieve the two versions of the resource addressed by bind variables `current_RESID` and `previous_RESID` after check-in.

You can cancel a check-out using PL/SQL function `DBMS_XDB_VERSION.unCheckOut`. Example 25-10 illustrates this.

Example 25-1 Creating a Repository Resource

```
VARIABLE targetPath      VARCHAR2(700)
VARIABLE current_RESID   VARCHAR2(32)
VARIABLE previous_RESID  VARCHAR2(32)

DECLARE
  res BOOLEAN;
BEGIN
  :targetPath  := '/public/t1.txt';
  IF (DBMS_XDB_REPOS.existsResource(:targetPath))
     THEN DBMS_XDB_REPOS.deleteResource(:targetPath);
  END IF;
  res := DBMS_XDB_REPOS.createResource(:targetPath, 'Mary had a little lamb');
END;
/
```

Example 25-2 Creating a Version-Controlled Resource

```
DECLARE
  resid DBMS_XDB_VERSION.RESID_TYPE;
BEGIN
  resid := DBMS_XDB_VERSION.makeVersioned(:targetPath);
  :current_RESID := resid;
  COMMIT;
```

[1] In Oracle XML DB, a version resource always has a *single* predecessor, that is, a single version that immediately precedes it. The WebDAV standard provides for the possibility of multiple predecessors.

```
END;
/
```

Example 25-3 Retrieving Resource Content by Referencing the Resource ID

```
SELECT XDBURIType(DBMS_XDB_REPOS.createOIDPath(:current_RESID)).getClob()
  FROM DUAL;

XDBURITYPE(DBMS_XDB_REPOS.CREATEOIDPATH(:CURRENT_RESID)).GETCLOB()
--------------------------------------------------------------------
Mary had a little lamb

1 row selected.
```

Example 25-4 Checking Out a Version-Controlled Resource

```
BEGIN
  DBMS_XDB_VERSION.checkOut(:targetPath);
  COMMIT;
END;
/
```

Example 25-5 Updating Resource Content

```
DECLARE
  content        BLOB;
  newContentBlob BLOB;
  newContentClob CLOB;
  source_offset  INTEGER := 1;
  target_offset  INTEGER := 1;
  warning        INTEGER;
  lang_context   INTEGER := 0;
BEGIN
  -- Lock the resource using a dummy update.
UPDATE RESOURCE_VIEW
  SET RES =
    XMLQuery('copy $i := $p1 modify
                (for $j in $i/Resource/DisplayName
                 return replace value of node $j with $p2)
                return $i'
               PASSING
                 RES AS "p1",
                 XMLCast(XMLQuery('declare namespace ns =
                                    "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                                   $r/ns:Resource/ns:DisplayName/text()'
                                  PASSING RES AS "r" RETURNING CONTENT)
                         AS VARCHAR2(128)) AS "p2"
               RETURNING CONTENT)
    WHERE equals_path(res, :targetPath) = 1;
  -- Get the LOB locator.
  SELECT XMLCast(XMLQuery('declare namespace ns =
                           "http://xmlns.oracle.com/xdb/XDBResource.xsd"; (: :)
                           $r/ns:Resource/ns:XMLLob'
                          PASSING RES AS "r" RETURNING CONTENT)
                 AS BLOB)
    INTO content FROM RESOURCE_VIEW
    WHERE equals_path(RES, :targetPath) = 1;
  -- Update the LOB.
  newContentClob := 'Hickory dickory dock, the mouse ran up the clock';
  DBMS_LOB.createTemporary(newContentBlob, false, DBMS_LOB.CALL);
  DBMS_LOB.convertToBlob(newContentBlob, newContentClob,
                         DBMS_LOB.getLength(newContentClob),
                         source_offset, target_offset,
```

```
                        nls_charset_id('AL32UTF8'), lang_context, warning);
  DBMS_LOB.open(content, DBMS_LOB.lob_readwrite);
  DBMS_LOB.trim(content, 0);
  DBMS_LOB.append(content, newContentBlob);
  DBMS_LOB.close(content);
  DBMS_LOB.freeTemporary(newContentBlob);
  DBMS_LOB.freeTemporary(newContentClob);
  COMMIT;
END;
/
```

Example 25-6 Checking In a Version-Controlled Resource

```
DECLARE
  resid DBMS_XDB_VERSION.RESID_TYPE;
BEGIN
  resid := DBMS_XDB_VERSION.checkIn(:targetPath);
  :previous_RESID := DBMS_XDB_VERSION.getPredsByRESID(resid)(1);
  COMMIT;
END;
/
```

Example 25-7 Retrieving Resource Version Content Using XDBURITYPE and CREATEOIDPATH

```
SELECT XDBURIType(DBMS_XDB_REPOS.createOIDPath(:current_RESID)).getClob()
  FROM DUAL;

XDBURITYPE(DBMS_XDB_REPOS.CREATEOIDPATH(:CURRENT_RESID)).GETCLOB()
--------------------------------------------------------------------
Mary had a little lamb

1 row selected.

SELECT XDBURIType(DBMS_XDB_REPOS.createOIDPath(:previous_RESID)).getClob()
  FROM DUAL;

XDBURITYPE(DBMS_XDB_REPOS.CREATEOIDPATH(:PREVIOUS_RESID)).GETCLOB()
---------------------------------------------------------------------
Hickory dickory dock, the mouse ran up the clock

1 row selected.
```

Example 25-8 Retrieving Resource Version Content Using GETCONTENTSCLOBBYRESID

```
SELECT DBMS_XDB_VERSION.getContentsCLOBByRESID(:current_RESID) FROM DUAL;

DBMS_XDB_VERSION.GETCONTENTSCLOBBYRESID(:CURRENT_RESID)
-------------------------------------------------------
Mary had a little lamb

1 row selected.

SELECT DBMS_XDB_VERSION.getContentsCLOBByRESID(:previous_RESID) FROM DUAL;

DBMS_XDB_VERSION.GETCONTENTSCLOBBYRESID(:PREVIOUS_RESID)
--------------------------------------------------------
Hickory dickory dock, the mouse ran up the clock

1 row selected.
```

Example 25-9 Retrieving Resource Version Metadata Using GETRESOURCEBYRESID

```
SELECT XMLSerialize(DOCUMENT DBMS_XDB_VERSION.getResourceByRESID(:current_RESID)
                    AS CLOB INDENT SIZE = 2)
  FROM DUAL;
<Resource xmlns="http://xmlns.oracle.com/xdb/XDBResource.xsd" Hidden="false"
          Invalid="false" VersionID="2" ActivityID="0" Container="false"
          CustomRslv="false" VersionHistory="false" StickyRef="true">
  <CreationDate>2009-05-06T12:33:34.012133</CreationDate>
  <ModificationDate>2009-05-06T12:33:34.280199</ModificationDate>
  <DisplayName>t1.txt</DisplayName>
  <Language>en-US</Language>
  <CharacterSet>UTF-8</CharacterSet>
  <ContentType>text/plain</ContentType>
  <RefCount>1</RefCount>
  <ACL>
    <acl description="Public:All privileges to PUBLIC"
         xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                             http://xmlns.oracle.com/xdb/acl.xsd"
         shared="true">
      <ace>
        <grant>true</grant>
        <principal>PUBLIC</principal>
        <privilege>
          <all/>
        </privilege>
      </ace>
    </acl>
  </ACL>
  <Owner>HR</Owner>
  <Creator>HR</Creator>
  <LastModifier>HR</LastModifier>
  <SchemaElement>http://xmlns.oracle.com/xdb/XDBSchema.xsd#text</SchemaElement>
  <Contents>
    <text>Mary had a little lamb</text>
  </Contents>
  <VCRUID>69454F2EF12E3375E040578C8A1764B5</VCRUID>
  <Parents>69454F2EF12F3375E040578C8A1764B5</Parents>
</Resource>

1 row selected.

SELECT XMLSerialize(DOCUMENT DBMS_XDB_VERSION.getResourceByRESID(:previous_RESID)
                    AS CLOB INDENT SIZE = 2)
  FROM DUAL;
<Resource xmlns="http://xmlns.oracle.com/xdb/XDBResource.xsd" Hidden="false"
          Invalid="false" VersionID="1" Container="false" CustomRslv="false"
          VersionHistory="false" StickyRef="true">
  <CreationDate>2009-05-06T12:33:34.012133</CreationDate>
  <ModificationDate>2009-05-06T12:33:34.012133</ModificationDate>
  <DisplayName>t1.txt</DisplayName>
  <Language>en-US</Language>
  <CharacterSet>UTF-8</CharacterSet>
  <ContentType>text/plain</ContentType>
  <RefCount>0</RefCount>
  <ACL>
    <acl description="Public:All privileges to PUBLIC"
         xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
```

```
                              http://xmlns.oracle.com/xdb/acl.xsd"
          shared="true">
       <ace>
         <grant>true</grant>
         <principal>PUBLIC</principal>
         <privilege>
           <all/>
         </privilege>
       </ace>
     </acl>
  </ACL>
  <Owner>HR</Owner>
  <Creator>HR</Creator>
  <LastModifier>HR</LastModifier>
  <SchemaElement>http://xmlns.oracle.com/xdb/XDBSchema.xsd#text</SchemaElement>
  <Contents>
    <text>Hickory dickory dock, the mouse ran up the clock</text>
  </Contents>
  <VCRUID>69454F2EF12E3375E040578C8A1764B5</VCRUID>
</Resource>

1 row selected.
```

Example 25-10 Canceling a Check-Out Using UNCHECKOUT

```
DECLARE
  resid DBMS_XDB_VERSION.RESID_TYPE;
BEGIN
  resid := DBMS_XDB_VERSION.unCheckOut(:targetPath);
END;
/
```