Tamino

Tamino WebDAV Functionality

Version 10.1

April 2018

WEBMETHODS

This document applies to Tamino Version 10.1 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: INS-TINSWEBDAV-101-20180413

Table of Contents

Preface

WebDAV is a general-purpose interface for accessing documents on a remote server. As a standard for collaborative web authoring, WebDAV is based on HTTP. Common procedures such as the creation, retrieval and deletion, as well as the organization of documents are supported. Users can edit web resources with the same ease as they can edit resources in a local directory. The WebDAV standard clearly makes its mark in application areas such as Instant Web publishing, Workgroups, Content Management, and Strategic File Management.

The Tamino WebDAV functionality allows communicating with Tamino via the WebDAV protocol. The following WebDAV functionality is supported:

RFC 2518 (HTTP Extensions for Distributed Authoring -- WEBDAV)
RFC 3253 (Versioning Extensions to WebDAV)
RFC 3744 (Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol)
WebDAV SEARCH

The Tamino WebDAV functionality treats all operations as atomic. This means, in regards of transactionality, that the Tamino WebDAV functionality provides no such thing as a session, or the capability of isolation, commit or rollback.

This documentation covers the following topics:

The WebDAV Standard
Basic Methods
Properties
Versioning
Security
Searching
WebDAV and Tamino

1 The WebDAV Standard

The WebDAV (Web-based Distributed Authoring and Versioning) standard is defined as an extension of HTTP/1.1 (**IETF RFC 2616**) allowing for additional methods and functionality which exceed the abilities of traditional HTTP. The WebDAV functionality implements the WebDAV extension of HTTP/1.1. Thereby the WebDAV functionality goes beyond Tamino's interactive interface and enables access to Tamino resources via common client interfaces which support the WebDAV standard. You can simply drag and drop Tamino content onto your desktop – no lengthy commands are needed. Popular applications such as Microsoft (TM) Web Folders, Web Browsers, and other standalone clients, as well as a growing number of common servers already support WebDAV. Clearly, Tamino and WebDAV are a perfect match.

The WebDAV standard covers an extended set of methods in addition to HTTP's abilities and facilitates XML to describe how these methods are communicated.

The WebDAV specification (**IETF RFC 2518**) covers major aspects which have proved to be of particular interest to distributed production environments:

- **Collections**
  In contrast to HTTP's URL resource access of a single file, WebDAV offers the concept of organizing any number of individual resources into WebDAV collections. A collection can be compared to a file system directory, providing efficient means for accessing and structuring resources. Managing collections and resources of remote repositories opens a vast field of new possibilities for authoring documents and tools in network environments.
- **Locking**
  WebDAV offers various concepts for collaboration on resources which may be accessed by any number of users simultaneously.
- **Properties**
  To help to make resources more valuable, WebDAV allows properties, or “metadata”, to be assigned to any type of data. This is where XML's inherent extensibility is particularly suited. Here again, the possibilities exceed the scope of helpful retrieval mechanisms, or references, by far, and leave adequate room for further developments.

- **Security (ACL)**
  "In distributed authoring scenarios resources may be accessible by multiple principals. To control how these principals can access and alter a resource, access controls are needed. These controls define what actions a particular principals is allowed to exercise on a particular resource." (from the ACL Standard)

  The ACL standard defines privileges on a resource basis. Each resource can be individually secured by different access rights.

- **Versioning and Configuration Management (DeltaV)**
  "Versioning, parallel development, and configuration management are important features for collaborative authoring of Web content. Version management is concerned with tracking and accessing the history of important states of a single Web resource, such as a standalone Web page. Parallel development provides additional resource availability in multi-user, distributed environments and lets authors make changes on the same resource at the same time, and merge those changes at some later date. Configuration management addresses the problems of tracking and accessing multiple interrelated resources over time as sets of resources, not simply individual resources. Traditionally, artifacts of software development, including code, design, test cases, requirements, help files, and more have been a focus of configuration management. Web sites, comprised of multiple inter-linked resources (HTML, graphics, sound, CGI, and others), are another class of complex information artifacts that benefit from the application of configuration management." (from the DeltaV Standard)

  The DeltaV standard defines a set of new WebDAV methods to allow versioning and configuration management over the internet.

- **WebDAV SEARCH**
  The WebDAV SEARCH standard defines a WebDAV method and a query language to search with a WebDAV namespace for resources based on Boolean expressions on properties and content.

2 Basic Methods

Tamino WebDAV supports the following WebDAV functionalities as defined in RFC 2518:

```
PUT
GET
HEAD
OPTIONS
DELETE
MKCOL
COPY
MOVE
PROPFIND
PROPPATCH
LOCK
UNLOCK
```

These HTTP methods are required to provide a view on documents as known from file systems.

Some of these functions already exist in Tamino (`PUT`, `GET`, `HEAD`, `OPTIONS`, `DELETE`) but are Tamino-specific methods whereas the methods listed above are specific to Tamino WebDAV.

In order to distinguish the methods used by the standard access protocol to Tamino (`DELETE`, `GET`, `POST`, `PUT`) and the methods used in WebDAV, a special identifier is added to the request that is supposed to use the WebDAV access protocol.

Example:

Standard Tamino `GET` request:

http://`<host:port>`*/tamino/*`<dbname>`*/*`<tamino collection>`*/doctype/*`<@inoid>`

WebDAV URL for a `GET` request:

http://`<host:port>`*/tamino/*`<dbname>`*/*`<identifier>`*/*`<resource path>`*/*`<resource name>`

where, as listed in the following table, the *<identifier>* consists of “ino:dav” and one of several predefined WebDAV collection names:

Identifier	Purpose
ino:dav/ino:dav	For standard resources such as directories or files.
ino:dav/ino:davHistory	For history and versions.
ino:dav/ino:davWork	For work and workspaces.
ino:dav/ino:davWorking	For working resources.
ino:dav/ino:davPrincipal	A virtual collection that contains WebDAV users and groups.

On Windows, where the colon character (":") is typically the separator between drive letter and folders, the use of the colon in the URL's path can lead to problems for some clients, for example when using the `net` command to map Tamino WebDAV as a network drive to a drive letter. To circumvent this problem, Tamino allows an alternative syntax without colons in the URL path. The colon between *host* and *port* in the URL is generally not a problem for clients. The alternative syntax is:

- http://*<host:port>*/tamino.MyDB.dav/*<resource path>*/*<resource name>*
- http://*<host:port>*/tamino.MyDB.davHistory/*<resource path>*/*<resource name>*
- http://*<host:port>*/tamino.MyDB.davWork/*<resource path>*/*<resource name>*
- http://*<host:port>*/tamino.MyDB.davWorking/*<resource path>*/*<resource name>*
- http://*<host:port>*/tamino.MyDB.davPrincipal/*<resource path>*/*<resource name>*

In order to use this URL syntax, a special entry in the Apache *httpd.conf* file needs to be set. Add the following lines, if they are not yet present in *httpd.conf*:

```
<LocationMatch /tamino.*>
SetHandler ino
</LocationMatch>
```

Note: The WebDAV standard does not define any semantic for the HTTP `POST` request. No `POST` method is allowed in the WebDav Scope.

3 Properties

One major benefit of storing documents in WebDAV compared to simple file systems is that each document (and folder) also contains properties. Some properties are provided by the WebDAV system, others are user-defined, thereby allowing applications to add any information they wish to the WebDAV resources.

Properties are basically key value pairs, where the key is a qualified name (`QName`) and the value is a string. Properties are represented as XML elements, where the key is the name of the element and the value is the content. This means that the restrictions of XML element names and content apply to WebDAV properties.

A property with the name `author` in the namespace *http://myhost.com/nspace1* with the value *John* could be represented as:

```
<ns1:author xmlns:ns1="http://myhost.com/nspace1">John</ns1:author>
```

WebDAV properties can also be XML fragments, where the content contains further XML elements.

The information in this chapter is organized under the following headings:

Live Properties

According to ***RFC 2518***, a live property is "a property whose semantics and syntax are enforced by the server". Please refer to ***RFC 2518***, ***RFC 3253***, and ***RFC 3744*** for a list of the available live properties. All the required live properties are supported by the Tamino. Some of the optional live properties are not available. The live properties are predefined and belong to the `DAV:namespace`.

An example of a live property is the content-type of a document, which has the name *getcontenttype* and would be represented as:

```
<d:getcontenttype xmlns:d="DAV:">test/xml</d:getcontenttype>
```

Dead Properties

According to ***RFC 2518***, a dead property is "a property whose semantics and syntax are not enforced by the server. The server only records the value of a dead property; the client is responsible for maintaining the consistency of the syntax and semantics of a dead property". This basically means that dead properties can be anything which can be represented as XML and which is not in the `DAV:namespace`, since that namespace is reserved for the live properties.

Special Tamino Properties

The Tamino Server offers some special properties which are used to control certain functional aspects. The special Tamino properties are in the namespace *http://namespaces.softwareag.com/tamino/response2*. These properties can only be applied to folders (WebDAV collections) and not to documents.

Collection

The property `collection` specifies the Tamino collection to which the WebDAV folder is mapped. By default, a folder is mapped to the collection `ino:dav`. If this property is set for a WebDAV folder, all documents stored in this folder and all subfolders are stored into the specified Tamino collection. Note that changing this property does not move the documents stored in the WebDAV folder to the new collection; only documents stored after the change are stored in the new Tamino collection. Note also that the collection to which the mapping is set must exist before specifying the mapping – it is not created automatically.

Since WebDAV is often used like a file system and should be able to store all kinds of documents, it is probably a good practice to set the mapped collections to optional schema usage, so that also documents without a matching schema can be stored to the respective WebDAV folder.

Non-XML Doctype

When a WebDAV folder is mapped to a Tamino collection, all non-XML data is stored into the doctype `ino:nonXML` by default. If another non-XML doctype should be used, it can be specified using the `non-xml-doctype` property.

Auto-Version Control

The Tamino Server provides a mechanism for putting WebDAV collections under auto-version control. This means that any document stored into that folder is automatically put under version control, and changes to the document can lead to automatically generated versions. To activate this option, the `auto-version-control` property can be set to any of the four support auto-versioning modes: checkout-checkin, checkout-unlocked-checkin, checkout, and locked-checkout. Set the value *`off`* to deactivate this option after it has been set.

Setting Properties

In most cases, the WebDAV properties will probably be set by using a WebDAV client or an application using a WebDAV API. When communicating directly with the Tamino, the properties are set using the `PROPPATCH` command. The following example uses `PROPPATCH` to set to *`mydoctype`* the doctype for non-XML documents stored in the WebDAV folder *`myfolder`*:

```
PROPPATCH /tamino/mydb/ino:dav/ino:dav/myfolder HTTP/1.1
Host: mymachine.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
  xmlns:Z="http://www.w3.com/standards/z39.50/">
  <D:set>
    <D:prop>
      <ino:non-xml-doctype
        xmlns:ino="http://namespaces.softwareag.com/tamino/response2">
        mydoctype
      </ino:non-xml-doctype>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

An example of a WebDAV client that supports the setting of properties and has been tested with Tamino is the DAV Explorer, an open source implementation from the University of California at Irvine (***http://www.davexplorer.org/***).

Limitations

The properties that can be stored for one WebDAV resource are limited in size. About 32 kilobytes of space are available for each resource – depending on the type of encoding used, this can be considerably less than 32,000 characters. The property values are indexed using standard indexes, so performing searches on the properties should be very efficient.

Accessing Properties from XQuery

The WebDAV properties are represented as XML fragments and can be accessed from within XQuery requests using special predefined functions. Please refer to the *Tamino WebDAV Functions* overview for details on these functions.

4 Versioning

The information in this chapter is organized under the following headings:

DeltaV Overview

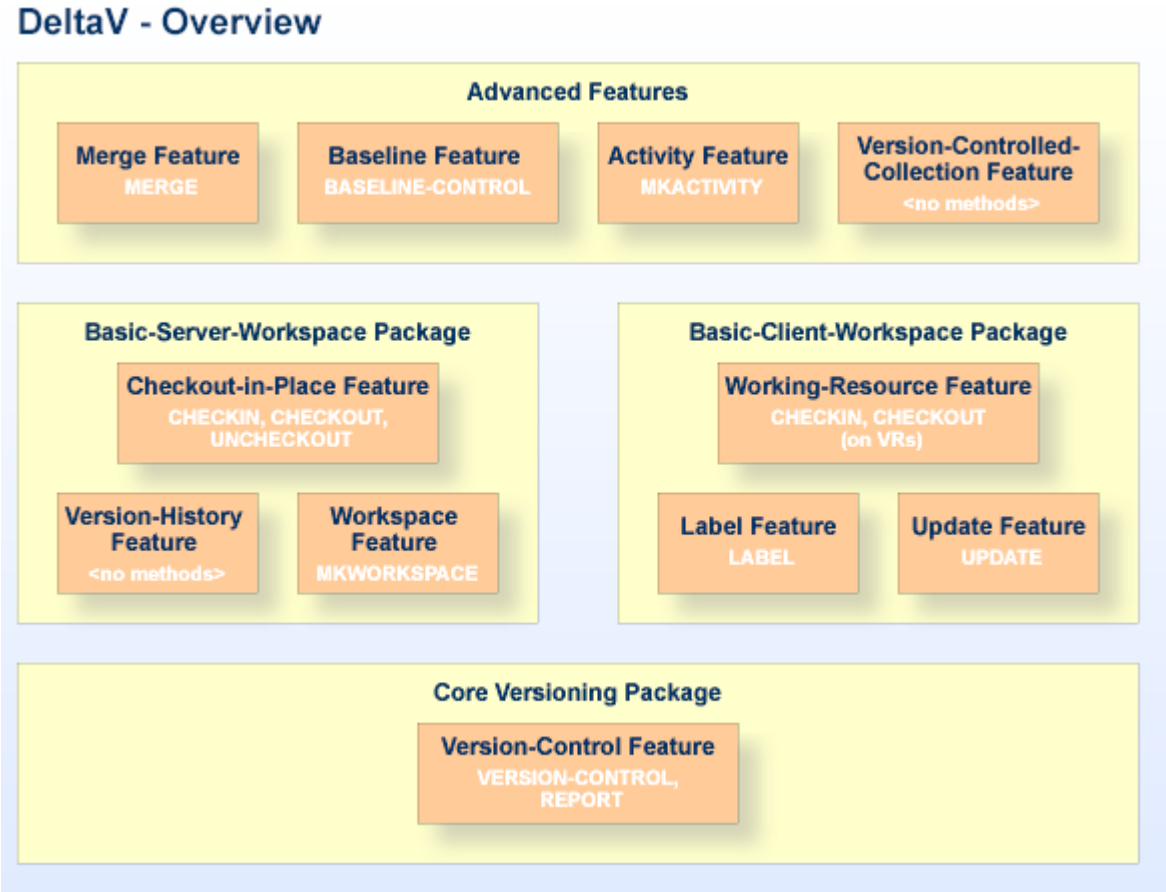
The web versioning and configuration management protocol DeltaV has been developed as an open standards-based infrastructure that supports collaborative development scenarios and includes:

- explicit versioning
- automatic versioning for versioning-unaware clients
- version history management
- workspace management
- baseline management
- activity management
- URL namespace versioning

The DeltaV protocol is an extension to the WebDAV (Web Distributed Authoring and Versioning) protocol, which itself extends the Hypertext Transfer Protocol (HTTP), the core network protocol that carries web traffic between a web server and a web browser.

Despite its name, the WebDAV protocol only provides facilities for remote collaborative authoring of documents and does not provide any versioning capabilities.

DeltaV itself is a network protocol (IETF RFC 3253 **http://www.ietf.org/rfc/rfc3253.txt**) that allows remote versioning and configuration management of documents that are stored in a web server. It defines four implementation levels, implementing 11 features.



Versioning in Tamino WebDAV

Tamino WebDAV provides a URL space in which all versions of an object are located.

For the default context of WebDAV the URL is as follows:

http://`<host:port>`*/tamino/*`<dbname>`*/ino:dav/ino:davHistory/*`<historyID>/<versionID>`

All objects are dealt with primarily as non-XML objects.

In the case of resources that are stored outside of the WebDAV context in a collection of the Tamino namespace (ino:`<collection>`), for instance in order to follow a schema, the convention enforced here is that for an object named for example *myCollection* a Tamino collection *v.myCollection* is expected. Should this dedicated location bearing the prefix "v." not exist then the creation of versions will fail.

Subversion Overview

Subversion is an open source version control system. For more details about Subversion, refer to **http://subversion.tigris.org/**.

Tamino's WebDAV versioning functionality can also be used via the Subversion protocol. This feature can be used to connect a Subversion client to a Tamino repository. After you have set up a Tamino Subversion repository, you can use a Subversion client to check files into and out of this repository.

We recommend the Eclipse Subversion client “Subclipse”. For details, refer to **http://subclipse.tigris.org/**.

To set up a Tamino Subversion repository

Choose a name that is not the same as the name of any existing WebDAV collection or Subversion repository in the referenced Tamino database.

- Issue the following command:

  ```
  http://<host>/tamino/<dbname>/ino:collection?_admin=ino:CreateSvnRepository("<name>")
  ```

  Example: Database name: "mydb"; repository name: "myrepos":

  ```
  http://localhost/tamino/mydb/ino:collection?_admin=ino:CreateSvnRepository("myrepos")
  ```

To access a Tamino Subversion repository from a Subversion client

- Issue a command of the following form:

  ```
  http://localhost/tamino/mydb/ino:dav/ino:dav/myrepos
  ```

To remove a Tamino Subversion repository

- When the Tamino Subversion repository is no longer needed, it can be removed by issuing the following command.

  Caution: The entire contents of the repository will be deleted.

```
http://<host>/tamino/<dbname>/ino:collection?_admin=ino:RemoveSvnRepository("<name>")
```

Basic Versioning Functions

Tamino WebDAV supports the following HTTP methods as defined in RFC 3253:

- VERSION-CONTROL
- CHECKIN
- CHECKOUT
- UNCHECKOUT
- REPORT

VERSION-CONTROL

The Version-Control feature provides support for putting a resource under version control, creating an associated version-controlled resource and version history resource. A server indicates that it supports the version-control feature by including the string "version-control" as a field in the DAV header in the response to an `OPTIONS` request.

The version-control feature must be supported if any other versioning feature is supported.

Putting a resource under version control

In order to track the history of the content and dead properties of a versionable resource, an author can put the resource under version control with a `VERSION-CONTROL` request. A `VERSION-CONTROL` request performs three distinct operations:

1. It creates a new version history resource (VHR). Each version history resource is assigned a new distinct and unique server-defined URL.
2. It creates a new version resource (VR) and adds it to the new version history resource. The body and dead properties of the new version resource are a copy of those of the versionable resource. The server assigns the new version resource a new distinct and unique URL.
3. It converts the versionable resource into a version-controlled resource (VCR). The version-controlled resource continues to be identified by the same URL that identified it as a versionable resource. As part of this conversion, it adds a `DAV:checked-in` property, whose value contains the URL of the new version resource.

It is often useful to have access to a version history even after all version-controlled resources for that version history have been deleted. A server can provide this functionality by supporting version history resources. A version history resource is a resource that exists in a server defined namespace and therefore is unaffected by any deletion or movement of version-controlled resources. A version history resource is an appropriate place to add a property that logically applies to all states of a resource. The `DAV:expand-property` report can be applied to the `DAV:version-set` of

a version history resource to provide a variety of useful reports on all versions in that version history.

CHECKIN

A CHECKIN request can be applied to a checked-out version-controlled resource to produce a new version whose content and dead properties are copied from the checked-out resource.

If a CHECKIN request fails, the server state preceding the request MUST be restored.

(Quoted from RFC 3253 - Versioning Extensions to WebDAV - **http://www.ietf.org/rfc/rfc3253.txt**.)

CHECKOUT

A CHECKOUT request can be applied to a checked-in version-controlled resource to allow modifications to the content and dead properties of that version-controlled resource.

If a CHECKOUT request fails, the server state preceding the request MUST be restored.

(Quoted from RFC 3253 - Versioning Extensions to WebDAV - **http://www.ietf.org/rfc/rfc3253.txt**.)

UNCHECKOUT

An UNCHECKOUT request can be applied to a checked-out version-controlled resource to cancel the CHECKOUT and restore the pre-CHECKOUT state of the version-controlled resource.

If an UNCHECKOUT request fails, the server MUST undo any partial effects of the UNCHECKOUT request.

(Quoted from RFC 3253 - Versioning Extensions to WebDAV - **http://www.ietf.org/rfc/rfc3253.txt**.)

REPORT

A REPORT request is an extensible mechanism for obtaining information about a resource. Unlike a resource property, which has a single value, the value of a report can depend on additional information specified in the REPORT request body and in the REPORT request headers.

(Quoted from RFC 3253 - Versioning Extensions to WebDAV - **http://www.ietf.org/rfc/rfc3253.txt**.)

Auto-Versioning

Additionally, in order to support access by simple clients such as an HTTP 1.1 client, auto-versioning is supported.

Ideally, as a document is edited using a WebDAV-capable client, the DeltaV server will automatically version its contents. DeltaV provides this capability, using a feature known as auto-versioning

There are two styles of auto-versioning, depending on whether a new version is created every time the resource is updated, or only when the resource is unlocked. In the first case, a `PUT/PROPPATCH` (this means a request to modify the state) is preceded by a `CHECKOUT`, and is followed by a `CHECKIN`. That is, a `PUT/PROPPATCH` is expanded to `CHECKOUT` > `PUT/PROPPATCH` > `CHECKIN`. This auto-versioning approach is ideally suited for HTTP clients.

Note: This auto-versioning behavior can be used with tools like XML Spy from Altova.

In the second auto-versioning style, a `LOCK` request also results in a `CHECKOUT`, and an `UNLOCK` additionally results in a `CHECKIN`. This works well for authoring clients that using locking, since they typically take out a lock at the start of an editing session, and remove the lock at the end, and hence auto-versioning based on locks causes the authoring session to automatically be bracketed by `CHECKOUT` and `CHECKIN` operations.

Note: This behavior is implemented for example with Microsoft Office 2000 products: `LOCK` with **open** und `UNLOCK` with **close**.

This auto-versioning approach is ideally suited for WebDAV clients using the WebDAV `lock` feature.

All resources have a live property called `DAV:auto-version` assigned. When a "non-DeltaV" client performs a `PUT` or `PROPPATCH` on a resource, the value of the `DAV:auto-version` property determines the behavior:

Property	Behavior
`<checkout-checkin/>`	Perform an implicit (auto-) `CHECKOUT` and `CHECKIN`.
`<checkout-unlocked-checkin/>`	`auto-CHECKOUT`, and wait for a lock to vanish before `auto-CHECKIN`.
`<checkout/>`	`auto-CHECKOUT`, and wait for a lock to vanish before `auto-CHECKIN` but wait for an explicit `CHECKIN`.
`<locked-checkout/>`	Require a lock. `LOCK` causes `auto-CHECKOUT`, `UNLOCK` causes `auto-CHECKIN`.

Advanced Versioning Functions

In addition to the basic versioning functions, Tamino WebDAV supports the following methods:

- UPDATE
- LABEL
- MKWORKSPACE

UPDATE

The Update feature provides a mechanism for changing the state of a checked-in version-controlled resource to be that of another version from the version history of that resource.

The `UPDATE` method modifies the content and dead properties of a checked-in version-controlled resource (the "update target") to be those of a specified version (the "update source") from the version history of that version-controlled resource.

The response to an `UPDATE` request identifies the resources modified by the request, so that a client can efficiently update any cached state it is maintaining. Extensions to the `UPDATE` method allow multiple resources to be modified from a single `UPDATE` request

LABEL

In order to identify various sources that belong to a certain version, labels can be issued.

Labels offer the ability to associate a name with a specific version of various resources stored. By issuing a meaningful symbolic name to a snapshot of the various resources it is possible to later recreate the end-result of these combined resources and thereby to create a version history.

A label can automatically be assigned by a server, or it can be assigned by a client in order to provide a meaningful name for that version. A given version label can be assigned to at most one version of a given version history, but client assigned labels can be reassigned to another version at any time. Note that although a given label can be applied to at most one version from the same version history, the same label can be applied to versions from different version histories.

For certain methods, if the request-URL identifies a version-controlled resource, a label can be specified in a Label request header to cause the method to be applied to the version selected by that label from the version history of that version-controlled resource.

A `LABEL` request can be applied to a version to modify the labels that select that version. The case of a label name is preserved when it is stored and retrieved. When comparing two label names to decide if they match or not, a server uses a case-sensitive URL-escaped UTF-8 encoded comparison. If a `LABEL` request is applied to a checked in version-controlled resource, the operation is applied to the DAV:checked-in version of that version-controlled resource.

MKWORKSPACE

A `MKWORKSPACE` request creates a new workspace resource. A server may restrict workspace creation to particular collections, but a client can determine the location of these collections from a `DAV:workspace-collection-set OPTIONS` request.

If a `MKWORKSPACE` request fails, the server state preceding the request must be restored.

(Quoted from RFC 3253 - Versioning Extensions to WebDAV - **http://www.ietf.org/rfc/rfc3253.txt**.)

Workspaces

When working with version-controlled resources it is convenient for the user to have a workspace where he can create and modify copies of such resources without the need to do a check-out-in-place. For this reason a dedicated location is provided:

http://`<host:port>`*/tamino/*`<dbname>`*/ino:dav/ino:davWork/*`<myWorkSpace>`

All objects stored in this location are treated as non-XML objects.

5 Security

The WebDAV Access Control Protocol is currently defined as an **IETF Standard (RFC3744)** and is an extension to the WebDAV protocol.

The WebDAV Access Control Protocol provides a facility for secure access control of content and properties under WebDAV control. Implementing a security mechanism can be just as simple as handling a file system.

In the WebDAV security context, users and groups are called "principals". Security is defined by access control lists (ACLs) on resources, where each ACL specifies relationships between principals and privileges that apply for that resource.

An ACL contains a set of access control entries (ACEs). Each ACE specifies a principal and a set of privileges that are either granted or denied to that principal. When a principal executes a WebDAV request on a resource, the server evaluates the ACEs in the ACL to determine if the principal has permission for that operation. If matching privileges are found which explicitly grant or deny the operation, they apply. If no matching privileges are found, the parent of the resource (the parent collection) is queried for privileges. If no matching privileges are found all the way up to the root folder, then the operation is denied.

The information in this chapter is organized under the following headings:

Privileges

Tamino supports the privileges defined in RFC 3744 and adheres to the defined aggregation rules. Besides the standard privileges, Tamino also provides an additional privilege "security" which is in the namespace "http://namespaces.softwareag.com/tamino/response2". The aggregation is shown in the following diagram:

```
DAV:all
  |
  +- DAV:read
  |
  +- ino:security
  |    |
  |    +- DAV:read-acl
  |    |
  |    +- DAV:read-current-user-privilege-set
  |    |
  |    +- DAV:write-acl
  |
  +- DAV:write
  |    |
  |    +- DAV:write-content
  |    |
  |    +- DAV:write-properties
  |    |
```

```
|   +- DAV:bind
|   |
|   +- DAV:unbind
|
+- DAV:unlock
```

The privilege names are qualified names (QNames). In the above diagram all privileges with prefix "DAV" are in the namespace "DAV:" and the prefix "ino" defines the above mentioned namespace.

Please refer to **RFC3744** for the exact semantics of the privileges. The `ino:security` privilege is simply an aggregation of all privileges related to ACL operations.

Behavior

When an object is accessed via Tamino WebDAV, the server supports checks that correspond to the access control list definitions found in WebDAV ACL (RFC 3744). This functionality is provided by the request method ACL.

The WebDAV functionality distinguishes between authenticated and unauthenticated users, who are assigned to the groups DAV:authenticated and DAV:unauthenticated.

When Tamino authentication is deactivated (i.e. the XML property "Authentication" is not set to "web server" or "none") then all users accessing Tamino are assigned to the group DAV:unauthenticated, so the privileges assigned to that group apply for all users.

Users and Groups

The WebDAV specifications do not define a new method for authenticating users. The mechanisms provided in HTTP are to be used.

Since Tamino allows for authenticating users via the HTTP basic authentication scheme, this mechanism is used for WebDAV access, as well as the users and groups known in the standard Tamino Security environment, which are also used for WebDAV security.

Resources are under control of access control lists (ACL) within Tamino, whereas access control elements (ACE) refer to a WebDAV principal.

Following the standard Tamino security model, the WebDAV principals are defined in the `ino:security` collection. The users are defined in `ino:security` in the doctype `ino:user`. The groups are defined inside `ino:group`.

Tamino also allows group definitions obtained from external authentication services (LDAP or the local operating system). The element `ino:group` allows the addition of user-defined properties,

which are stored inside the element `ino:properties`. These properties can be accessed and modified when executing `PROPFIND` and `PROPPATCH` on the principal URLs.

All principals in the WebDAV context are addressed via a URL. The following URL scheme applies when addressing users and groups:

For Users:

http://<host:port>/tamino/<dbname>/ino:dav/ino:davPrincipal/ino:user/<userid>

where the *<userid>* corresponds to the `ino:userid` attribute of the user definition in `ino:user`

For Groups:

http://<host:port>/tamino/<dbname>/ino:dav/ino:davPrincipal/ino:group/<groupname>

where the *<groupname>* corresponds to the `ino:groupname` attribute of the group definition in `ino:group`.

Domains

In order to address users or groups associated to domains, the syntax appended to the URLs is illustrated in the following example:

http://<host:port>/tamino/<dbname>/ino:dav/ino:davPrincipal/ino:user/<domainid>\<userid or groupname>

Unsupported Features and Special Behavior

The WebDAV ACL specification defines some optional functionality and also some variations, which are left at the disposal of the implementation of the server. The following section describes such special behavior:

- Principal Resources
- DAV:group Property
- DAV:acl-restrictions/no-invert

- REPORT

Principal Resources

The WebDAV ACL specification defines that principal resources are to be exposed via URLs. However, none of the WebDAV methods are required to be supported on such URLs.

The Tamino server allows for sending `PROPFIND` requests on principal resources and also on principal collections.

`PROPPATCH` is supported on a few of the properties.

No other methods are allowed in principal resources. Attempts to use them will result in a 405 HTTP response code (Method Not Allowed).

DAV:group Property

The `DAV:group` property always returns an empty result, since sources in Tamino do not have a group membership. This is in conformance to the WebDAV ACL specification.

DAV:acl-restrictions/no-invert

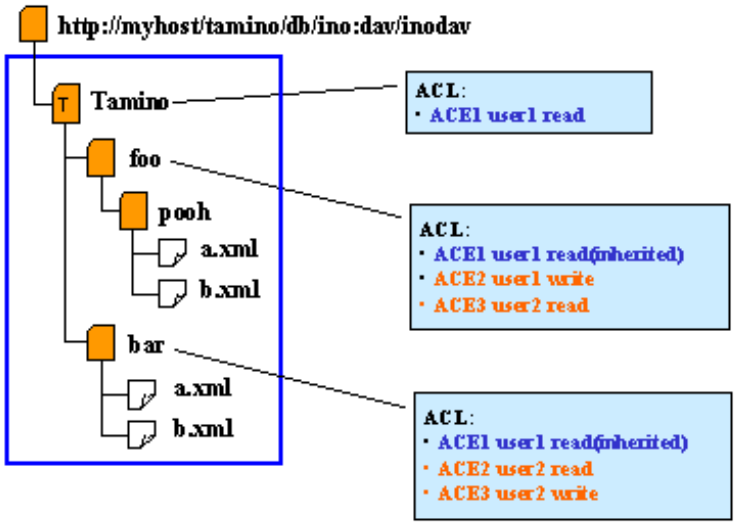
The Tamino server does not support the invert functionality on ACEs. The `DAV:acl-restrictions` property returns <no-invert>.

REPORT

The `acl-principal-prop-set` report is not implemented in the Tamino server. The evaluation of this report would in many cases result in major parts of the database having to be read, which could make the costs for executing such a report unreasonably high.

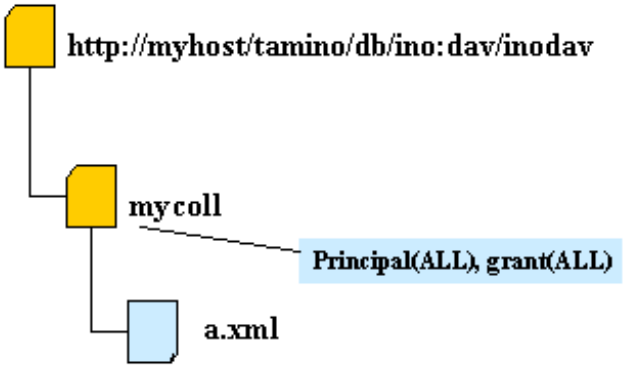
Access Control Lists: Examples

Examples

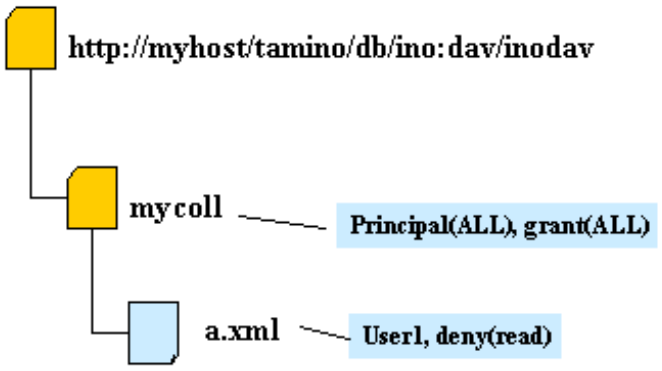


This example demonstrates that rights on certain properties can be inherited. A child node inherits the properties of its parent nodes and can contain further properties.

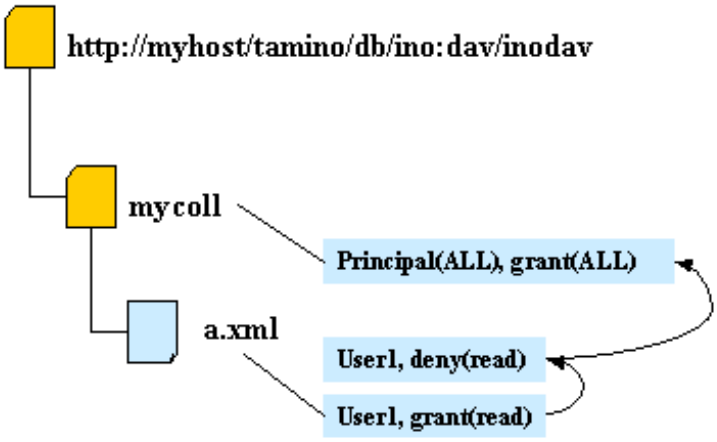
The following graphics demonstrate how access rights are granted. Arrows indicate the processing order.



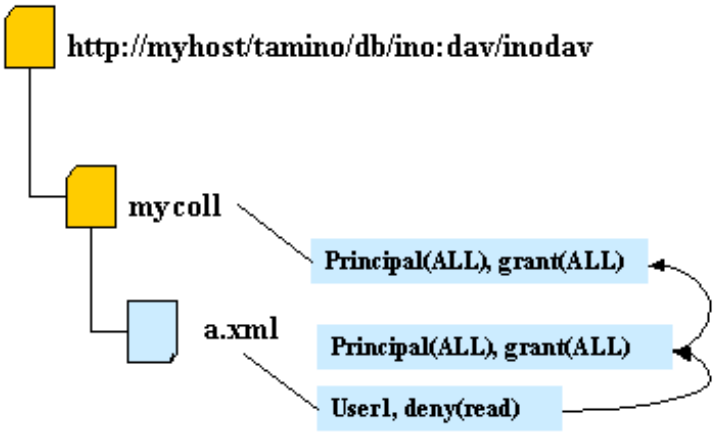
All users can read and write the resource *a.xml*, caused by the inherited ACE of the root folder.



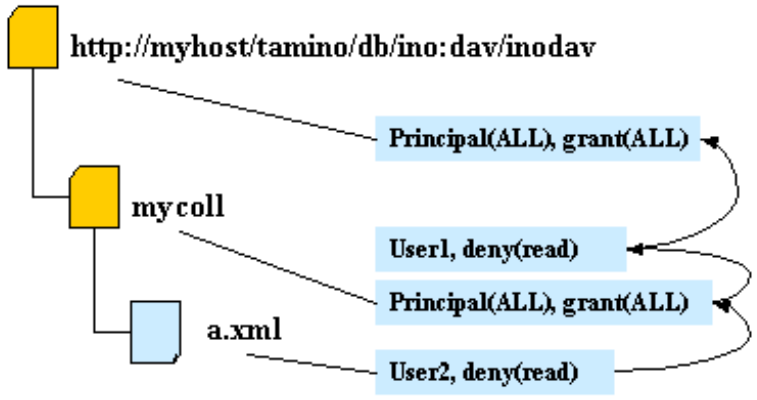
User1 is not allowed to read the resource *a.xml*. All other users can read this resource.



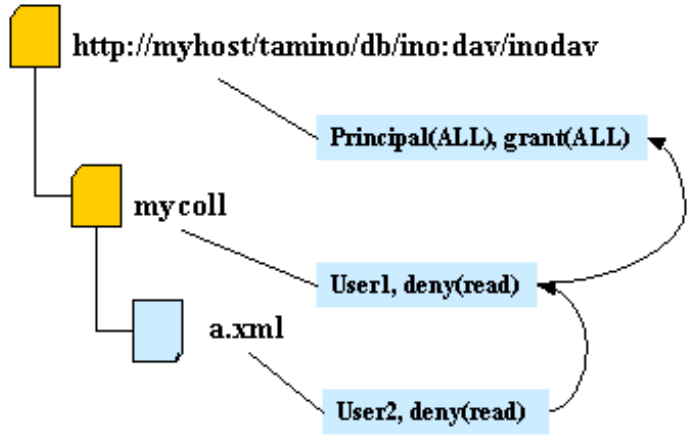
User 1 is allowed to read resource *a.xml*.



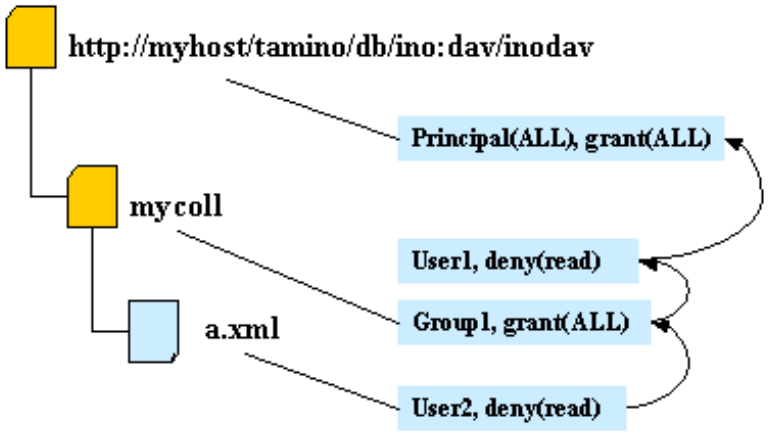
User1 is not allowed to read resource *a.xml* since the deny(read) is found before the grant(ALL) can be evaluated. As soon as a privilege is found which either grants or denies the requested operation, the ACE evaluation is finished and all subsequent ACEs are not evaluated.



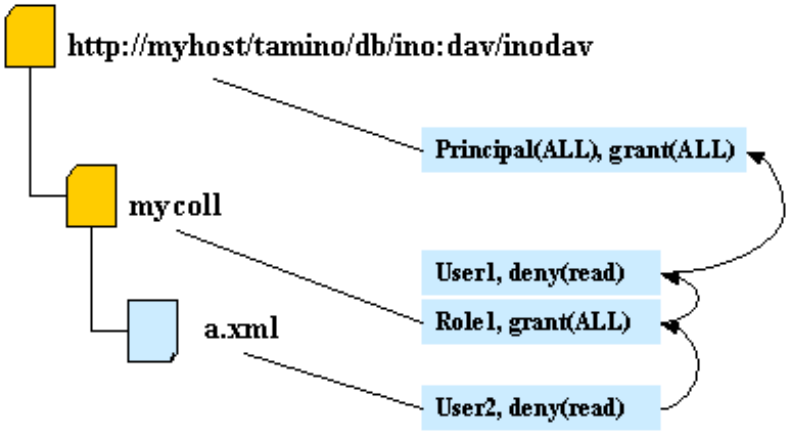
User 1 is allowed to read resource *a.xml*. User 2 is not allowed to read resource *a.xml*.



User1 and User2 are not allowed to read resource *a.xml* and User3 is allowed to read resource *a.xml*.



User 1 is a member of Group 1: User1 is allowed to read the resource *a.xml*. User 2 is not allowed to read resource *a.xml*.



User 1 owns the role Role 1: User 1 is allowed to read resource *a.xml*. User 2 is not allowed to read resource *a.xml*.

For further information and background knowledge on this topic, we suggest visiting the following link:

- **http://www.ietf.org/**

6 Searching

Tamino supports the WebDAV SEARCH grammar (the WebDAV SEARCH specification formerly known as DASL), which specifies a search grammar known as `DAV:basicsearch`. The specifications of DASL and WebDAV SEARCH are available at **https://tools.ietf.org/**.The results of the `SEARCH` request method are the same as the results of a `PROPFIND` with the requested properties.

The client invokes the `SEARCH` method on a resource that will perform the search (the search arbiter) and includes a text/xml request entity that contains the query. The search arbiter performs the query. The search arbiter sends the results of the query back to the client in the response. The server sends a text/xml entity that matches the `PROPFIND` response.

WebDAV SEARCH is an application of HTTP/1.1. It is a lightweight search protocol for transporting queries and result sets. It allows clients to make use of server-side search facilities. WebDAV SEARCH minimizes the complexity of clients, thus facilitating widespread deployment of applications capable of utilizing the WebDAV SEARCH mechanisms which allows to negotiate a query language.

The WebDAV SEARCH standard defines the `SEARCH` method, the WebDAV SEARCH response header, the `DAV:searchrequest` XML element, and the `DAV:basicsearch` XML element and query grammar. The basic query language element enables searching for the existence or value of a property and searching for a substring in a resource content.

DAV:basicsearch Grammar

The `basicsearch` grammar provides an SQL-like query language (select ... from ... where ... order by ...) for selecting WebDAV documents (URLs) and optional associated WebDAV properties. The "where" clause restricts the result set by a boolean filter, where the boolean elements are WebDAV property comparisons, for example `getContentLength > 1024`.

If such a query is executed within the namespace of a Tamino store, the query is internally compiled into a corresponding Tamino XPath statement and directly executed in Tamino, obtaining optimum execution performance.

Example:

Find all *mpeg* resources in the *mycoll* store and display the URLs and the properties `getcontenttype` and `getcontentlength`.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<searchrequest xmlns:D="DAV:">
  <D:basicsearch>
    <D:select>
      <D:prop>
        <D:getcontentlength/>
        <D:getcontenttype/>
      </D:prop>
    </D:select>
    <D:from>
      <D:scope>
        <D:href>/ino:dav/mycoll/</D:href>
      </D:scope>
    </D:from>
    <D:where>
      <D:eq>
        <D:prop>
          <D:getcontenttype/>
        </D:prop>
        <D:literal>audio/mpeg</D:literal>
      </D:eq>
    </D:where>
  </D:basicsearch>
</searchrequest>
```

Response:

```
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>/ino:dav/mycoll/sound/cathgirls.mp3</D:href>
    <D:propstat>
      <D:prop>
        <D:getcontentlength>471040</D:getcontentlength>
        <D:getcontenttype>audio/mpeg</D:getcontenttype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  ...
  <D:response>
    <D:href>/ino:dav/mycoll/sound/camarillobrillo.mp3</D:href>
    <D:propstat>
      <D:prop>
        <D:getcontentlength>919928</D:getcontentlength>
        <D:getcontenttype>audio/mpeg</D:getcontenttype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

7 WebDAV and Tamino

Storing resources in Tamino XML Server is a different paradigm than storing resources using WebDAV. However, complete transparency between both approaches is achieved by additional information, which is required in case:

- a document is stored via WebDAV and should become part of a Tamino collection of documents;
- a document is stored via the standard Tamino interface and should be exposed to the WebDAV file system view.

Access from WebDAV to Tamino

All objects in the WebDAV context are stored in the default WebDAV collection as non-XML. This provides maximum compliance with the WebDAV standard, i.e. without Tamino-specific handling of XML documents.

However, this default behavior can be changed:

A WebDAV directory's default storage location can be changed using a `PROPPATCH` command with the property `<Tamino namespace>:<collection>`. A newly-created resource (`PUT`, `COPY`) thus inherits the Tamino collection property and is stored to the specified Tamino collection. Note that this applies only to newly-stored objects.

Access from Tamino to WebDAV Properties

Access to WebDAV Properties

Access to WebDAV properties is provided by a set of Tamino XQuery functions:

```
tdf:resource
tdf:isDescendantOf
tdf:getProperty
tdf:getProperties
```

For more information on these and other XQuery functions, please refer to the *XQuery 4 Reference Guide*.

Index