



# WEBADM ADMINISTRATOR GUIDE

The specifications and information in this document are subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. This document may not be copied or distributed by any means, in whole or in part, for any reason, without the express written permission of RCDevs Security.

WebADM and OpenOTP are trademarks of RCDevs. All further trademarks are the property of their respective owners.

No guarantee is given for the correctness of the information contained in this document. Please send any comments or corrections to [info@rcdevs.com](mailto:info@rcdevs.com).

# WebADM Administrator Guide

[OpenLDAP](#) [Novell](#) [Active Directory](#) [Web-Application](#) [Web-Service](#) [Token](#) [U2F](#) [Proxy User](#) [Push](#) [Yubikey](#) [SMS](#)

## 1. Product Documentation

This document is a configuration guide for RCDevs WebADM. The reader should notice that this document is not a guide for configuring WebADM applications (Web Services and WebApps). Specific application guides are available through the RCDevs online documentation library. WebADM installation and setup is not covered by this guide and is documented in the [RCDevs WebADM Installation Guide](#).

## 2. Product Overview

WebADM is a powerful Web-based LDAP administration software designed for professionals to manage LDAP Organization resources such as domain users and groups. It is also the configuration interface for RCDevs Web Services and WebApps (end-user applications).

WebADM usage is 100% graphical and many features are documented inside the management interface itself. Moreover, WebADM has been built for a maximum ease of use and its usage is very intuitive. For this reason, not all the features are documented in this guide as they are most of the time self-explanatory.

WebADM can be used standalone, as a powerful LDAP management interface. It provides a hierarchical view of LDAP Organizations, SQL and file-based audit trails and ultra-rich LDAP object management features. It is the centralized administration interface for all RCDevs Web services and Web applications. It supports domains of users, LDAP groups, multi-level applications' policies, web service client applications' access control rules... The possibilities for managing your enterprise security are nearly unlimited and WebADM's flexibility makes it possible to implement any enterprise security requirement.

WebADM is compatible with Novell eDirectory, Microsoft Active Directory 2008 and later, OpenLDAP, Apple OpenDirectory, Oracle/Sun Directory and RCDevs Directory Server. Other directories might work but are not tested nor supported by RCDevs. WebADM can manage and federate all your organization directories in one single interface. It connects your Active Directory, Novell, OpenLDAP all together and provides a hierarchical view, delegated administration and powerful management for your directories. With OpenOTP, it implements your centralized authentication system, working with your existing directories and domains.

WebADM understands both Microsoft Active Directory domains and UNIX PAM-LDAP users. You can seamlessly manage both systems from the interface. Better, WebADM can extend your Active Directory users (with POSIX functionalities) so that they work with your PAM-LDAP UNIX systems. WebADM is also the only software which able to unify your Microsoft and UNIX infrastructure.

WebADM does not use static LDAP object administration templates. Instead, it is able to read and understand any LDAP directory schema. With this information, it is able to provide dynamic administration interfaces for managing existing objects with their attributes and create new ones. To achieve this, WebADM includes a set of objectclass and attribute specifications providing information for manipulating specific data types. That means, when you connect a WebADM to an LDAP directory, it will read the LDAP server schemas and will immediately be able to manage the directory objects, without needing specific configurations or new object manipulation templates.

WebADM is also able to manipulate Unix, Windows accounts, groups and whatever data your directory is able to store, without

additional configurations. WebADM supports delegated administration and fine-grained access control to LDAP resources. Administrators can be created at different levels of the tree structure, with different privileges and views. WebADM includes all the necessary features to create new administrators, assign them quotas or settings, restrict tree access, etc... With Novell eDirectory, WebADM takes advantage of the LDAP built-in permissions (ACLs). Administrators can create new contexts with sub-administrators and assign them rights in the limit of their own authorizations, but without compromising the directory security.

WebADM can be used as a central management system for multiple LDAP trees. With the OptionSets, it provides a very simple way to assign settings to specific contexts. And, in order to provide even more detailed management policies, the OptionSets work with inheritance, so the settings can be re-defined into sub-contexts, or quota can be affined at sub-tree levels...

WebADM uses client certificates as default and recommended authentication mechanism for more security when administering LDAP resources. It includes its own PKI subsystem to create, renew, distribute and revoke user/server certificates. It also provide an OCSP endpoint for certificate revocations checks by client systems.

### Warning

Starting from WebADM version 1.4.2, any high availability and clustering feature require an RCDevs Enterprise license. Without a valid license file, the HA and cluster features are automatically disabled.

LDAP Server1 (RCDevs Directory) ↻

RCDevs Directory (3)

- dc=WebADM
- o=Demos
- o=Root (6)
  - cn=admin
  - cn=admins\_group
  - cn=marcus
  - cn=ppolicy
  - cn=user1
  - cn=user2
- ★ Create / Search  
Details / Check
- ★ Create / Search  
Details / Check

## WebADM Freeware Edition v2.0.7

Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home Admin Create Search Import Databases Statistics Applications About Logout

Hello Admin (*cn=admin,o=root*)

Connected as **Super Administrator** to 3bcd8b34772 ℹ

---

**Support Services**

License status: **Valid** (Cloud-based)

Maintenance included: **No**

Maintenance mode: **Disabled** ([Enable maintenance mode](#))

---

**Application Status**

MFA Authentication Server: **Ok** (v1.5.3)

Shared Session Server: **Ok** (v1.0.11)

SMS Hub Server: **Ok** (v1.2.0)

SSH Public Key Server: **Ok** (v2.0.9)

Administration Help Desk: **Ok** (v1.0.4)

OpenID & SAML Provider: **Ok** (v1.4.1)

Secure Password Reset: **Ok** (v1.1.0)

User Self-Service Desk: **Ok** (v1.2.0)

User Self-Registration: **Ok** (v1.2.0)

---

**Configurations Objects**

User Domains: <b>6</b> ( <a href="#">Details</a> )	Mount Points: <b>3</b> ( <a href="#">Details</a> )
Client Policies: <b>2</b> ( <a href="#">Details</a> )	Access Devices: <b>1</b> ( <a href="#">Details</a> )
Option Sets: <b>2</b> ( <a href="#">Details</a> )	Admin Roles: <b>2</b> ( <a href="#">Details</a> )

---

**Context & Permissions**

Administration Level: **Expert**

Login Context: **o=root** ([Details](#))

Tree Root Context: **Auto**

Created Objects: **All**

Allowed Configs: **All**

Allowed Databases: **All**

Managed Databases: **All**

Allowed Logfiles: **All**

Applied Option Sets: **o=root** ([Details](#)) ([Edit](#))

---

**Login Context Options**

Unicity Context: **o=root**

WebADM Quotas: **Disabled**

Figure 1a. WebADM Home Page (Virtual Appliance - RCDevs Directory Server)

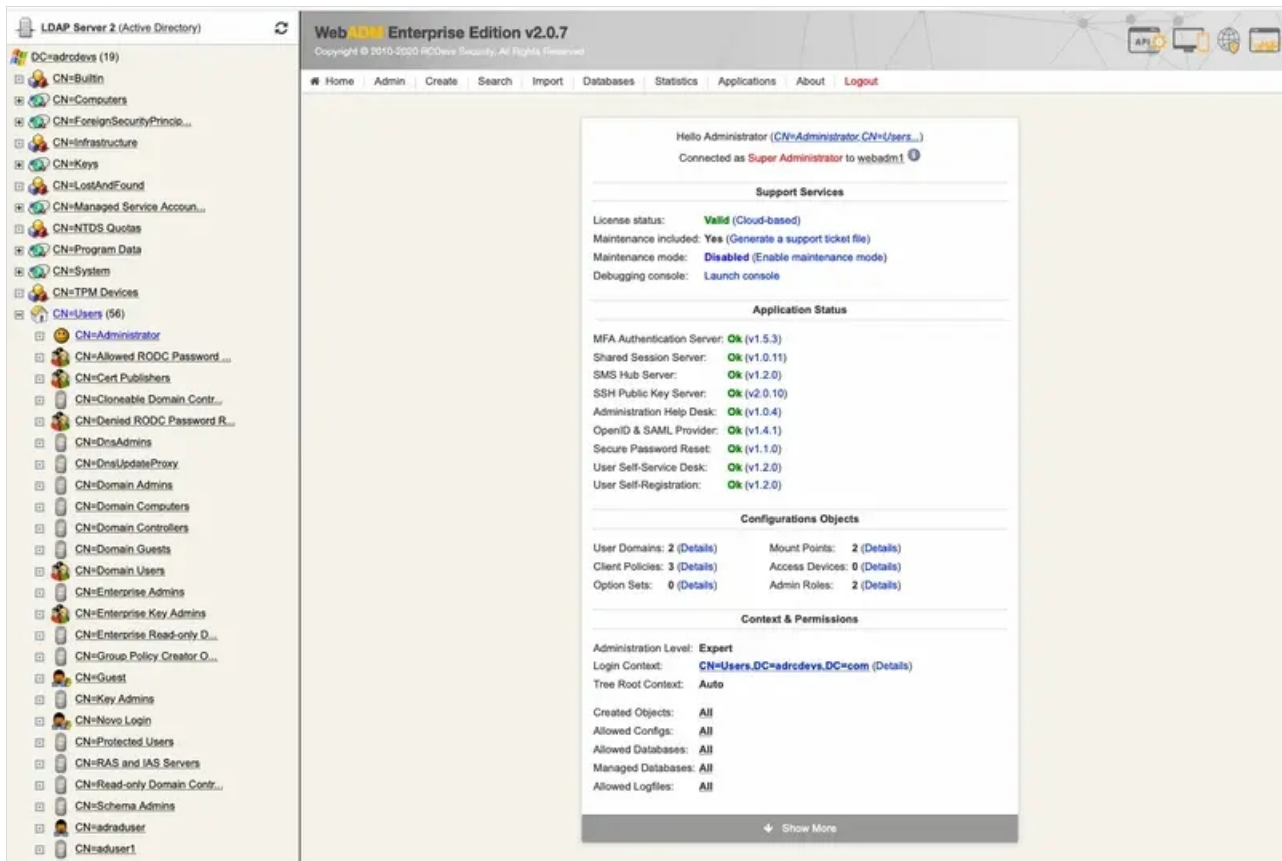


Figure 1b. WebADM Home Page (Virtual Appliance - Active Directory)

### 3. Files and Folders

Find below the WebADM software installation file structure and important files.

- › `/opt/webadm/bin/` : Location for WebADM service binaries and startup scripts.
  - › `webadm`: WebADM executable control script for starting and stopping the server processes. To start WebADM from a command line, issue `./webadm start`. To stop WebADM, issue `./webadm stop`.
  - › `setup`: Initial WebADM setup script run by the self-installer. The setup can be re-run manually at any time. The 'setup slave' command provides a slave mode setup for clustered environments. Please look at the WebADM High Availability documentation for details.
- › `/opt/webadm/doc/` : Location for WebADM documentation resources.
  - › `/opt/webadm/doc/scripts/`: This folder contains some useful scripts such as the tool for creating and renewing WebADM SSL certificates.
- › `/opt/webadm/conf/` : Location for WebADM configuration files.
  - › `webadm.conf` : Main configuration file. Defines the basic WebADM startup parameters, the location of WebADM-specific LDAP containers, WebADM proxy-user account DN, etc... Please look at Appendix A for an example of `webadm.conf` file with explanations.
  - › `objects.xml` : XML configuration file that defines the LDAP objects supported by WebADM and their related parameters. You can edit the XML definitions in this file to customize many aspects of the WebADM behavior.
  - › `servers.xml` : XML configuration file that specifies the server connections for LDAP, SQL, Session Server, PKI, SMTP and

HTTP proxies. Please look at Appendix B for an example of the servers.xml file with explanations.

- > `rsignd.conf`: PKI server (Rsignd) configuration file. Defines the integrated certificate authority settings and its clients. Please look at Appendix C for an example of `rsignd.conf` file with explanations.
- > `webadm.env`: Some runtime environment variables can be re-defined in this file. Please look at the bin/webadm startup script for the list of variables and the syntaxes. The following variables can be set:

- `INTERFACE`: The IP address the HTTP services must listen on.
- `HTTP_PORT_STD`: The HTTP unsecured port used for Administrator Portal and WebApps. This port is not used and is a redirection to the `HTTP_PORT_SSL` port.
- `HTTP_PORT_SSL`: The HTTP SSL port used for Administrator Portal and WebApps.
- `SOAP_PORT_STD`: The HTTP unsecured port used for the Web Services.
- `SOAP_PORT_SSL`: The HTTP SSL port used for the Web Services.
- `CACHE_MEMSIZE`: The amount of memory allocated to the WebADM shared caches.
- `REDIS_MEMSIZE`: The amount of memory allocated to the WebADM session manager service (ie. the local Redis instance).
- `REDIS_NOSYNC`: Set to Yes to disable session server replication over the cluster.
- `SSL_PROTOCOL`: For example: 'export SSL\_PROTOCOL="ALL -SSLv2 -SSLv3 -TLSv1 -TLSv1.1"' will disable SSLv2, SSLv3, TLSv1, TLSv1.1 and only allow TLSv1.2.

- > `license.key`: The license file (if any) provided by RCDevs or its partners for WebADM Enterprise use.
- > `/opt/webadm/websrvs/`: Location for WebADM Web Services. Applications are provided with self-installers and are automatically installed in this place.
- > `/opt/webadm/webapps/`: Location for WebADM Web Applications. Applications are provided with self-installers and are automatically installed in this place.
- > `/opt/webadm/lib/`: Location for WebADM system libraries.
- > `/opt/webadm/libexec/`: Location for WebADM system executables.
- > `/opt/webadm/logs/`: Location for log files produced by all the WebADM services.

The log files in WebADM are:

- > `webadm.log`: This is the main WebADM log file which includes general startup errors, Administrator Portal events, Manager API events, WebApps' events, Web Services' events. Any Web Service API including SOAP, JSON, JON-RPC, REST, XML-RPC logs its events to this log file.
- > `sessiond.log`: This log file contains the session server errors (i.e. the local Redis instance errors and warnings).
- > `rsignd.log`: This log file contains the PKI server events (both errors and client requests).
- > `watchd.log`: This log file contains the WebADM connector status heal check errors.
- > `/opt/webadm/temp/`: Location for WebADM temporary data files. Under this directory, you will find service PID files, socket files, Redis database dump files, license cache and license token.
- > `/opt/webadm/pki/`: Location for WebADM PKI server files and SSL certificate(s). This folder contains the WebADM CA signing certificate and key under the 'ca' sub-folder. The WebADM SSL certificate and key file used by the HTTP and Rsignd

services are `webadm.crt` and `webadm.key`. Custom certificate and key used for HTTPS access to port 443 (WebADM Admin portal and `webapps`) are also stored here (`custom.crt` and `custom.key`).

- > WebADM automatically checks the configuration files for syntax errors or mistakes and writes any problem discovered in the log file `/opt/webadm/logs/webadm.log` or directly in the startup script output.

WebADM configuration files are documented inline. Please look at the appendixes in this document for the default configuration files with comments.

## 4. WebADM Components

The WebADM server is composed of several server components and Web Portals, bundled into one unique application. These components include:

### 4.1 Network Services

#### 4.1.1 HTTP Server

WebADM provides only Web-based user interfaces and includes its own HTTP server which provides the administrator portal, the WebApp portal and a Web Services information portal. By default, all the Web interfaces are running over HTTPS on port 443.

The Web server includes a high performance multithreaded caching system which uses shared memory for maximum service responsiveness.

#### 4.1.2 SOAP Server

WebADM registered applications provide SOAP/XML interfaces only. The SOAP server component provides the HTTP and HTTPS listeners over which the SOAP/XML interfaces are accessible. By default, the SOAP service is running on HTTP port 8080 and HTTPS port 8443. The SOAP server includes a high performance multithreaded caching system which uses shared memory for maximum service responsiveness.

#### 4.1.3 Session Manager

Most of the WebADM registered applications require storing session data, timers, counters and object locks. The WebADM session manager provides those functionalities in a very high performant and distributed way. Best of all, a cluster of WebADM servers is always connected to a single session manager at-a-time for keeping any work data synchronized. This ensures the clustered systems are not affected by some kind of replay attacks and are able to handle the failover and load-balancing in the best conditions.

#### 4.1.4 Watchd Server

WebADM  $\geq$  v1.4 includes a new daemon called `watchd` which is responsible for checking the server connector statuses in real-time. `Watchd` permanently tests the connections for all servers declared in `conf/servers.xml` and informs WebADM about the current selection(s). With `Watchd` service running, your high-availability WebADM cluster is more efficient than ever for dealing with automatic connectors' failover and dead-peer detection. The `watchd` service is activated only when WebADM is running with an Enterprise license.

#### 4.1.5 PKI Server

WebADM includes its own PKI system for issuing user certificates. The PKI functionalities are used by the administrator portal and some WebADM applications. For security requirements, the PKI is working in client-server mode and the signing server does not run under the same system user than the other WebADM services. This ensures the Certificate Authority (CA) component cannot be accessed even through a breach in the other components.

As for the session manager, a clustered system should use only one PKI server for maintaining the coherence in the certificate serial numbers.

#### 4.1.6 Services Start and Stop

The WebADM startup script (webadm) is located in the `bin/` directory. Use the commands `webadm start`, `webadm stop` and `webadm restart` to start, stop or restart the WebADM services. The startup script is responsible for starting and stopping the WebADM HTTP server, the SOAP server, the session manager server and the PKI server. WebADM administration action logs are accessible in the Databases menu in WebADM. System logs are accessible in the `logs/webadm.log` file.

## 4.2 Web Portals

### 4.2.1 The Administrator Portal

This portal allows administrators to manage the LDAP objects of the organization, setup and configure WebADM applications. It provides a tree view of the LDAP organization, an object editor and many wizard-based LDAP operations. The administrator portal is accessible at the URL: `https://yourserver/`.

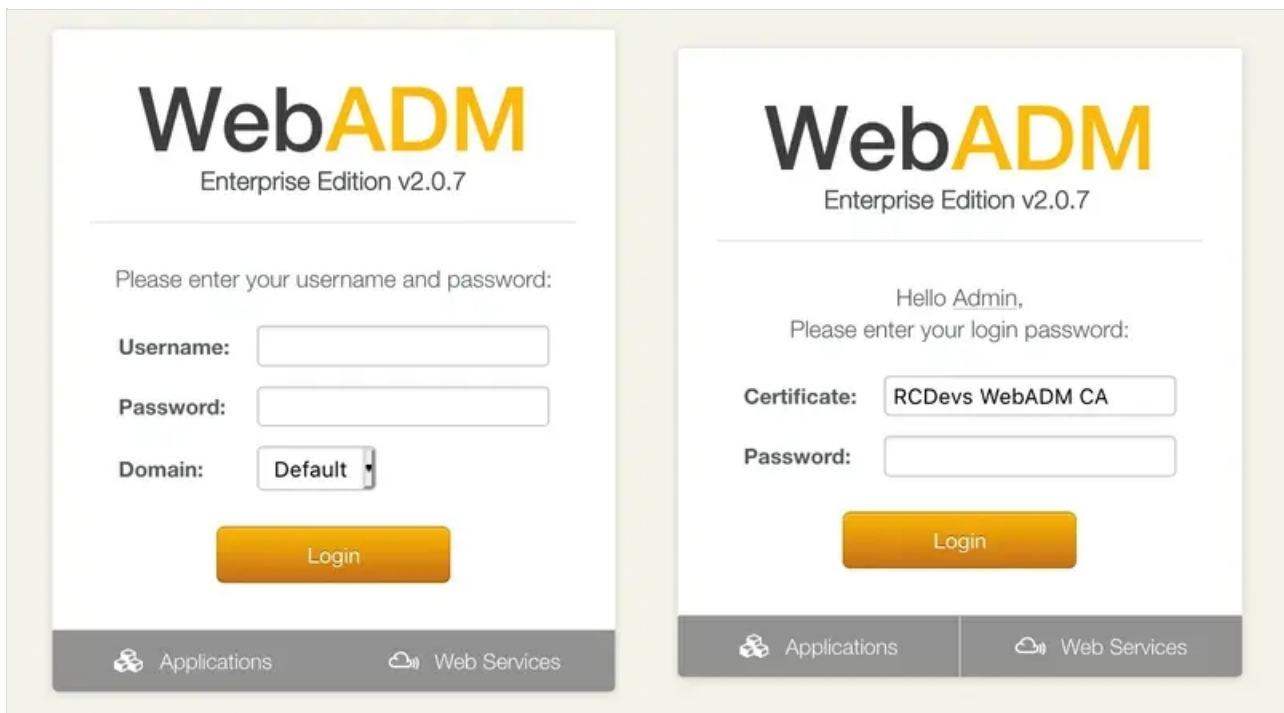


Figure 1. WebADM Admin Portal Login



Figure 2. WebADM Menu

The main Admin Portal menu at top of the page gives instant access to:



1. The Administrator Home Page: This page displays a summary of the administrator user details, installed applications, administrative options and tree options for the LDAP login context (see sections AdminRoles and OptionSets for details). When logging in as a super administrator (see WebADM main configuration file `conf/webadm.conf`), and if WebADM is not properly installed, the home page displays the button to access the initial setup wizard.

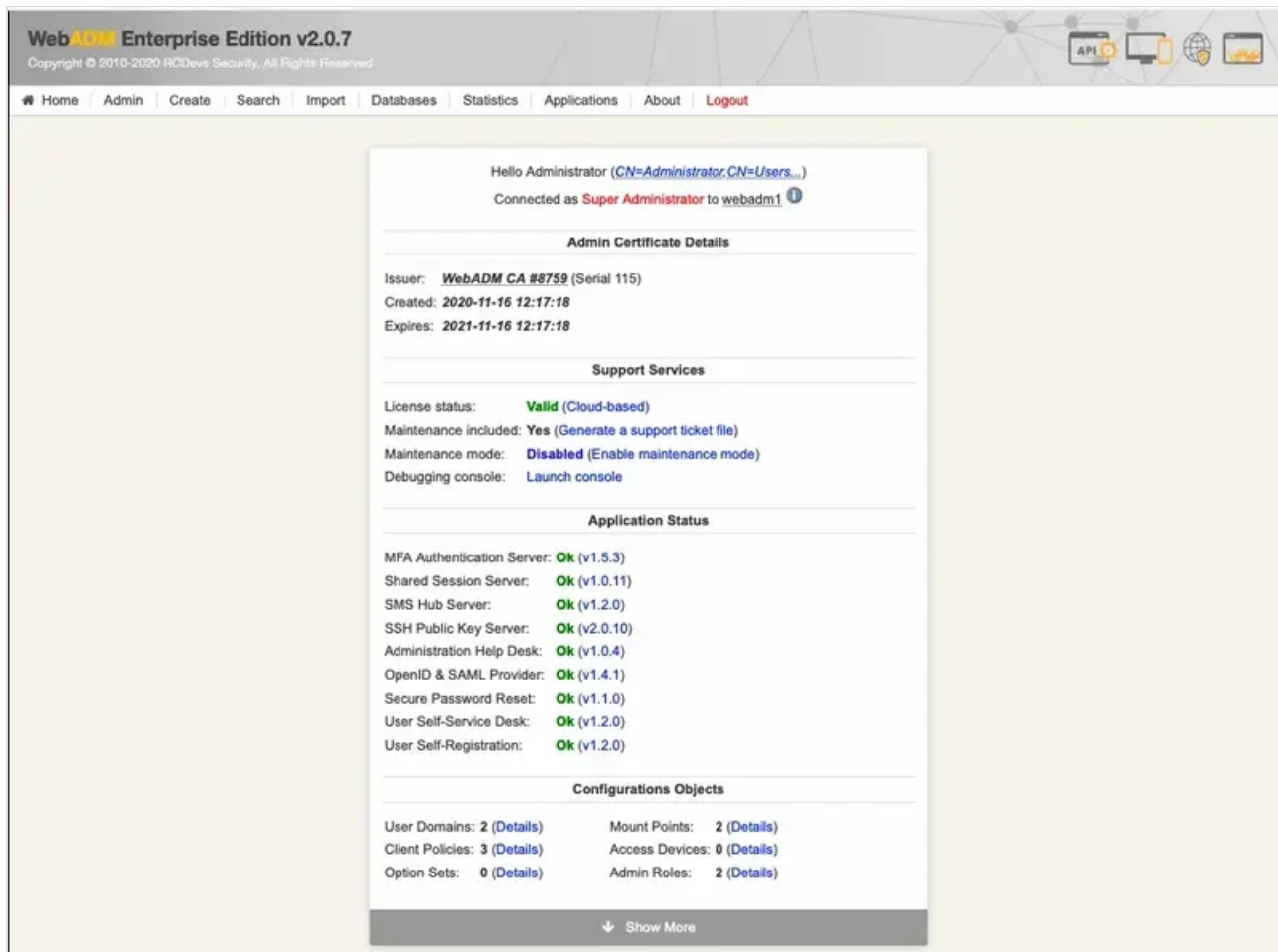


Figure 3. WebADM Home Page

2. The General Admin Menu: It displays a list of buttons for getting information concerning the LDAP server and schema, WebADM configurations, registered Domains, MountPoints and applications. It includes buttons for retrieving the Certificate Authority (CA) public certificate and the server SSL certificate, as well as buttons for flushing the WebADM caches.

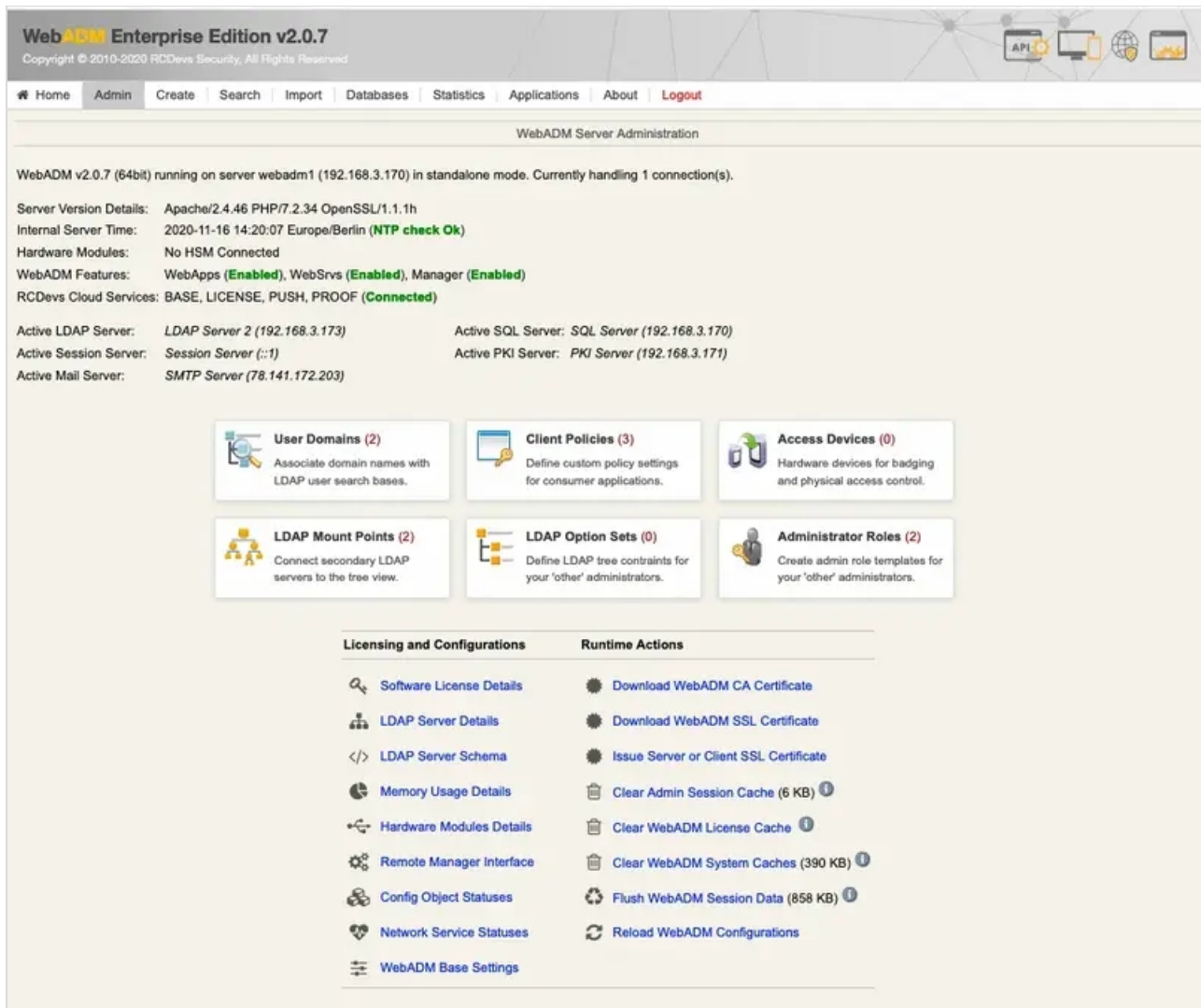


Figure 4. Admin Page

3. **The Object Creation Menu:** It displays a list of object that can be created. The listed objects are those which are specified in the object specifications file (`conf/objects.xml`). See section Creating Objects for details. The first object of the list is a generic LDAP object used for creating WebADM configuration objects such as MountPoints, AdminRoles, OptionSets, Domains, WebApps and Web Services. It includes a drop-down list for selecting one of these object types.
4. **The Search Menu:** It allows searching for LDAP objects. See section Searching Objects for details.
5. **The Import Menu:** It allows importing LDAP objects in batch using LDIF or CSV file formats. See section Importing Objects for details.
6. **The Databases Menu:** It displays the list of SQL log tables and application localized message tables. See section Log Viewer and Localized Messages Editor for details.
7. **The Application Menu:** It displays the list and status of the registered WebADM WebApps and Web Services.
8. **The About Menu:** It displays WebADM version information, changelog and some RCDevs contact email addresses.

**WebADM Enterprise Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security. All Rights Reserved

API | [Device Icons]

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

**WebADM Enterprise Edition**  
Version 2.0.7 (64bit) built October 22 2020

**RCDevs**  
security solutions

Provided by RCDevs Security SA  
Copyright © 2010-2020 RCDevs Security  
Copyright - Change Log - Release Notes  
Problem submission: [support@rcdevs.com](mailto:support@rcdevs.com)  
Commercial information: [sales@rcdevs.com](mailto:sales@rcdevs.com)  
General information: [info@rcdevs.com](mailto:info@rcdevs.com)

The installation and usage of this copy of WebADM is authorized by RCDevs provided that you fully comply with the WebADM copyright and usage terms.  
Please contact RCDevs' sales for any legal questions or issues.

Figure 5. About Page

#### 4.2.2 The WebApps Portal

The WebApp portal displays the list of registered applications with the links for directly entering them. This portal is accessible at the URL: <https://yourserver/webapps/>.

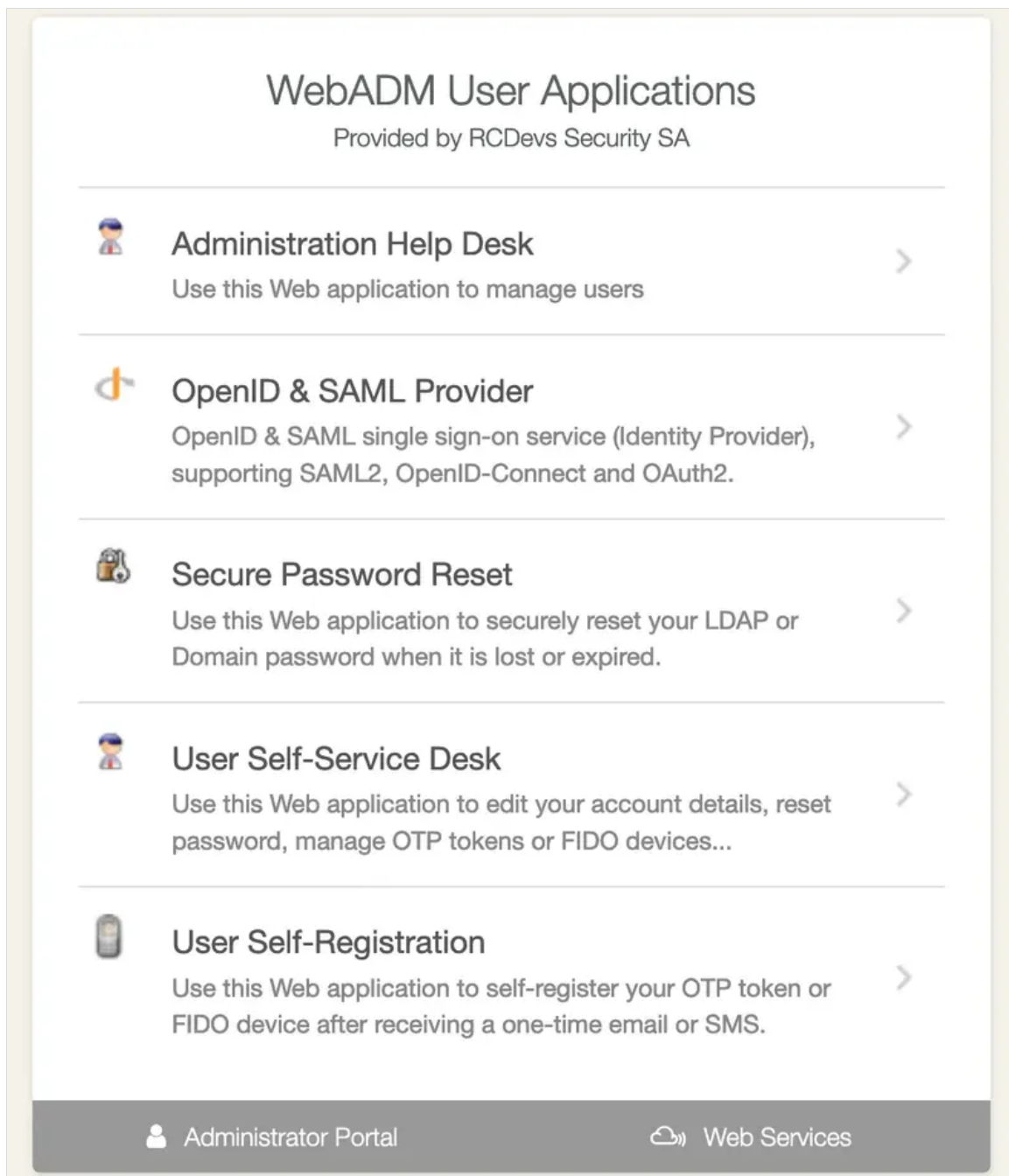


Figure 6. WebApps Portal

A WebApp named mywebapp can be accessed directly at the URL: <https://yourserver/webapps/mywebapp>.

The [/webapps/](#) HTTP location contains all the necessary resources for running a WebApp (meaning stylesheets and image references). That means there exist no references pointing to other locations on the Web server when you access a WebApp. The WebApps URL can also easily be placed behind a reverse-proxy which redirects URLs only for the WebApps location. This can be useful if you want to expose the WebApps over the Internet but not the admin portal.

#### Note

If you run a system with multiple servers, the admin portal can be disabled too in the WebADM main configuration file (`conf/webadm.conf`).

### 4.2.3 The Web Services Portal

This portal is only informational and displays the list of registered Web Services with their service descriptions files (WSDL). The Web Services portal is accessible at the URL: <https://yourserver/websrvs/>.

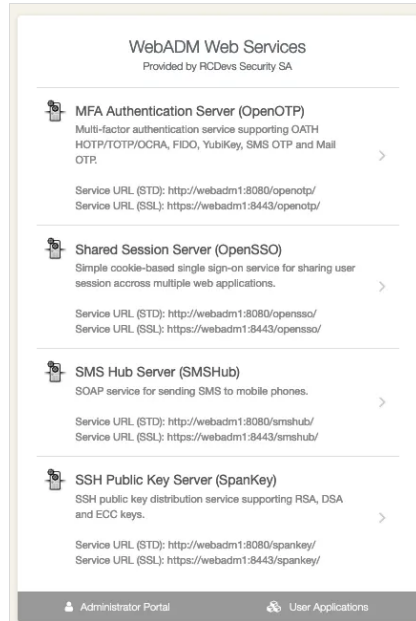


Figure 7. Web Services Portal

## 4.3 Web Services

RCDevs solutions (example: OpenOTP) run on top of the WebADM Server. The solutions are generally composed of both Web Services and end-user Web Applications (WebApps). WebADM is a container (application server), which embeds the HTTP and SOAP engines required by Web Services and WebApps.

A WebADM Web Service is a pluggable component to be installed (deployed) in WebADM. The Web Services provide final functionalities such as user authentication services. The Web Services provide:

1. A SOAP XML interface.
2. A WSDL service description file.
3. A graphical configurator.

You can review the list of registered Web Services and their status by categories in the Application Menu.

**WebADM Enterprise Edition v2.0.7**  
 Copyright © 2010-2020 RCDevs Security, All Rights Reserved


API, Desktop, Mobile, Security, Analytics

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

Categories

- ✓ Authentication (2)
- SMS Relay (1)
- Self-Service (4)
- Signature (1)
- Single Sign-On (2)

### Web Services

 **MFA Authentication Server (OpenOTP) v1.5.3 (Commercial)**

Multi-factor authentication service supporting OATH HOTP/TOTP/OCRA, FIDO, YubiKey, SMS OTP and Mail OTP.

Latest Version: 1.5.3 (Ok)

Status: **Enabled** [CONFIGURE] [REMOVE]


Service URL (SSL): <https://webadm1:8443/openotp/>

Service URL (STD): <http://webadm1:8080/openotp/>

Mobile Endpoint: <https://webadm1/ws/openotp/>

U2F Facet Endpoint: <https://webadm1/ws/appid/>

SOAP WSDL File: [openotp.wsdl](#)

 **SSH Public Key Server (SpanKey) v2.0.10 (Commercial)**

SSH public key distribution service supporting RSA, DSA and ECC keys.

Latest Version: 2.0.10 (Ok)

Status: **Enabled** [CONFIGURE] [REMOVE]

Service URL (SSL): <https://webadm1:8443/spankey/>

Service URL (STD): <http://webadm1:8080/spankey/>

U2F Application ID: <https://webadm1/ws/appid-ssh/>

SOAP WSDL File: [spankey.wsdl](#)

Figure 8. Registered Web Services

## 4.4 WebApps

A WebADM Web Application (WebApp) is a pluggable component to be installed (deployed) in WebADM. WebApps are generally companion application for some Web Services. For example, RCDevs OpenOTP Software Token requires the end users to register their secret Token keys, resynchronize their token application, etc... The Web Applications provide:

1. Some public Web pages.
2. Optional authentication with PKI, or Domain login (depending on the WebApp purpose).
3. A graphical configurator.

You can review the list of registered Web Services and their status in the Application Menu.

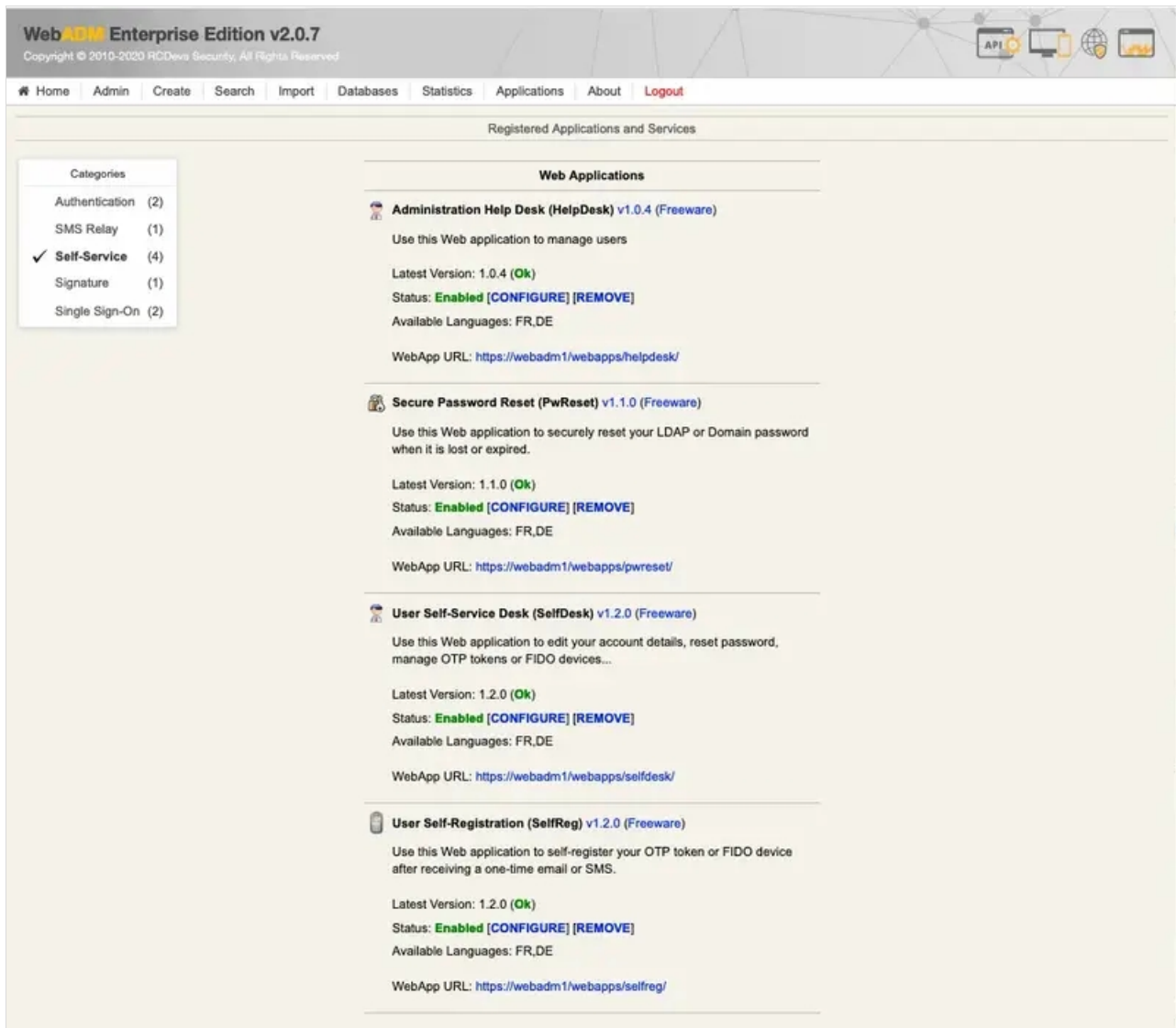


Figure 9. Registered Web Applications

RCDevs offer multiple WebApps for many purposes like:

- > Reset LDAP password,
- > Renew user certificate,
- > Register an SSH key,
- > Register a Token,
- > e-Sign a document and much more...

## 4.5 The Manager Interface

The Manager is a remote procedure call (RPC) interface which provides access to some WebADM user management functions and operations exported by your registered applications. The Manager also allows external systems such as Web portals to remotely trigger user management operations and actions from the network.

The Manager interface is accessible at the URL: <https://yourserver/manag/>. Please look at the section using the Manager Interface for details about the Manager Interface.

## 5. Configuration Files

The configuration files are self-documented. Please read them as part of this documentation.

The following settings are part of the main WebADM configuration file (`conf/webadm.conf`).

- > `admin_auth` : Administration Portal's authentication mode which can be:
  - > PKI: WebADM requires a client certificate and a login password.
  - > DN: WebADM requires a login DN and a password.
  - > UID: WebADM requires a domain name, a login name and a password.
  - > OTP: Like UID with an OTP challenge.
  - > U2F: Like UID with a FIDO-U2F challenge
  - > MFA: Like UID with both OTP and FIDO-U2F challenge

Using certificates is the most secure login method. To use certificate login, you must log in WebADM and create a login certificate for your administrators.

### Note

The UID mode requires a WebADM domain to exist and have its User Search Base set to the subtree where are located the administrator users. When using UID and if there is no domain existing in WebADM, the login mode is automatically forced to DN. You will also need to log in with the full user DN and set up a WebADM domain to be able to use the UID, OTP or U2F login mode.

- > `list_domains` : Show the domain list in a drop-down list in when `auth_mode` is set to UID, OTP or U2F.
- > `default_domain` : When `auth_mode` is set to UID, OTP or U2F, this defines the default domain when left blank. If `list_domains` is enabled, the default domain is pre-selected.
- > `manager_auth` : Manager API's authentication method. Only UID, PKI and DN are supported here. If you set the `admin_auth` with multi-factor (PKI, OTP or U2F), then you must either use `manager_auth` PKI or UID with a list of allowed client IPs (see below).
- > `manager_clients` : Optional list of client IPs which are allowed to use the Manager API. When `admin_auth` is configured with multi-factor and `manager_auth` is set to UID, then this client list is mandatory.
- > `proxy_user` : The proxy user is used by WebADM for accessing LDAP objects over which the administrator user does not have read permissions, or to access the LDAP resources out of an administrator session. The proxy user should have read permissions on the whole LDAP tree, and write permissions on the users and groups used by the WebApps and Web Services. A well-configured proxy user is mandatory for WebADM to work correctly.

Be sure to respect your directory password complexity policy for the proxy user password and to have the SSL enabled. Else, WebADM will not be able to create the proxy user during the graphical setup.

- > `super_admins` : Super administrators have extended WebADM privileges such as setup permissions, additional operations



and unlimited access to any LDAP encrypted data. Access restriction configured in the WebADM OptionSets and AdminRoles does not apply to super admins. You can set a list of individual LDAP users or LDAP groups here. With ActiveDirectory, your default administrator account should be something like `cn=Administrator,cn=Users,dc=mydomain,dc=com`. And you can replace the sample super\_admins group on the second line with an existing security group. WebADM administrators do not necessarily need to be domain admins. Fine grained permissions can be configured.

- > `container_ouclasses` : List of LDAP object classes to be considered by WebADM as LDAP containers.
- > `user_ouclasses` : List of LDAP object classes to be considered by WebADM as LDAP users. `user_ouclasses` is used to build the LDAP search filter when `auth_mode` is set to Domain. If your super administrator user does not have one of these objectclasses, then be sure to add one of its object classes to the list.
- > `group_ouclasses` : List of LDAP object classes to be considered by WebADM as LDAP groups.
- > `webadm_account_ouclasses` : List of LDAP object classes (extensions) to be considered by WebADM as WebADM account objects. WebADM accounts are usable by Web Services and WebApps.
- > `webadm_group_ouclasses` : List of LDAP object classes (extensions) to be considered by WebADM as LDAP groups with WebADM settings. Group settings are usable by Web Services and WebApps.
- > `webadm_config_ouclasses` : List of LDAP object classes to be considered as WebADM configuration objects.
- > `ignored_attrs` : List of LDAP attributes to be ignored by WebADM when creating or copying objects. This is a requirement for managing Microsoft ActiveDirectory users and groups.
- > `adminroles_containers`, `optionsets_container`, `webapps_container`, `websrvs_container`, `domains_container`, `clients_container` : WebADM containers required by WebADM for storing configuration objects. You have to change the container locations to match your LDAP tree base and constraints.
- > `session_timeout` : Here, you can set the timeout (in seconds) of a WebADM session. Sessions will be closed after this period of inactivity.
- > `cache_timeout` : Here, you can set the WebADM internal cache timeout.
- > `languages` : List of languages to be supported by your WebADM applications. The languages are used by the WebADM localized messages editor and for editing LDAP language attributes.
- > `encrypt_data` : Set to Yes if you want WebADM to encrypt LDAP sensitive data such as passwords, keys and session manager sessions with the AES-256 algorithm.
- > `encrypt_key` : This is the encryption key(s). The encryption key(s) must be 256bit base64-encoded random binary data. Use the command 'openssl rand -base64 32' to generate a new encryption key.

### Warning

If you change the encryption key, any encrypted data will become invalid!

You can set several encryption keys for key rollout. All the defined keys are used for decrypting data. And the first defined key is used to (re-)encrypt data. Features are automatically disabled.

- > `encrypt_mode` : WebADM provides 2 methods for user data, application settings and inventory data encryption:

1. The standard encryption (Standard): This is the default encryption mode when you set `encrypt_data` to Yes. In this mode, any

sensitive data is encrypted with the WebADM encrypt key. The encryption uses AES-256 in CBC block cipher mode and PKCS#7 padding. It is resistant to LDAP object copy out of WebADM.

2. The advanced encryption (Advanced): This mode is similar to the Standard mode but the encryption works per-object. Any encrypted data can also not be copied from one LDAP object to another. Also, LDAP objects cannot be moved or renamed out of WebADM without breaking the encryption.

- > `encrypt_hsm`: Set to Yes to enable HSM hardware encryption. The sensitive user data and other critical data will be encrypted with the hardware encryption module(s) defined with the `hsm_model` and `hsm_keyid` settings below. Setting No disables the hardware-based encryption of any updated data but does not prevent existing data (encrypted with the HSM) to be decrypted. You can also set it to No in order to switch from hardware encryption mode back to software encryption mode.
- > `hsm_model`: WebADM supports hardware security modules. When enabled, the hardware-based security complements the WebADM default software encryption: very sensitive user data like Token secrets or inventory data are transparently encrypted by the connected HSM(s) whereas other (less sensitive) data are encrypted using WebADM software encryption. WebADM currently supports Yubico's YubiHSM. Several YubiHSM modules can be used concurrently (in failover and load-balanced mode). Moreover, the addition or removal of HSM modules is hot-plug.
- > `hsm_keyid`: Like with the software encryption, multiple HSM key IDs (i.e. key handles) can be used concurrently and the rollout of a new AES hardware master key is supported. You can set several encryption key IDs for automatic key rollout. All the defined keys are used for decrypting data. And the first defined key is used to (re-)encrypt data.
- > `data_store`: It is now possible to choose the data storage mechanism to be used for storing user data and settings. By default, WebADM stores any user and group metadata in the LDAP objects. By setting the data store to 'SQL', these metadata are stored in a dedicated SQL table. LDAP data store remains the preferred option because it maximizes the system consistency. SQL data store should be used only if you need read-only LDAP access for the `proxy_user`.
- > `group_mode`: The group mode defines how WebADM will handle LDAP groups.
  - > Direct mode: WebADM finds user groups using the `memberof_attrs` defined above. In this case, the group membership is defined in the LDAP user objects.
  - > Indirect mode: WebADM finds user groups by searching group objects which contain the user DN as part of the `member_attrs`.
  - > Auto: Both direct and indirect groups are used.
  - > Disabled: All LDAP group features are disabled in WebADM.

By default, (when `group_mode` is not specified) WebADM handles both group modes.

- > `ldap_cache`: LDAP cache increases a lot of performances under high server loads. The cache limits the number of LDAP requests by storing resolved user DN and group settings. When enabled, results are cached for 300 secs.
- > `ldap_routing`: LDAP routing enables LDAP to request load-balancing when multiple LDAP servers are configured in `servers.xml`. You should enable this feature only if the LDAP server load becomes a bottleneck due to a large number of users (ex. more than 10000 users).
- > `ldap_uidcase`: Set to Yes if you need to handle LDAP login names with case sensitivity. By default, LDAP login names are case-insensitive.
- > `enable_admin`, `enable_webapps`, `enable_websrvs`, `enable_manager`: You can optionally disable main

WebADM features if you run multiple WebADM servers for different purposes. For example, if you don't want to provide the Administrator Portal on an Internet-exposed WebApps and Web Services server. By default, all the functionalities are enabled.

- > `log_format` : Format of the WebADM log file ( `/opt/webadm/logs/webadm.log` ). Set to CEF to enable Common Event Format logs to be used with Splunk servers.
- > `log_syslog` : Enables syslog logging (disabled by default). When enabled, WebADM system logs (any event in `webadm.log`) are sent to both the WebADM log files and the syslog.
- > `syslog_format` : Format of the WebADM syslog events. Set to CEF to enable Common Event Format logs to be used with Splunk servers.
- > `syslog_facility` : Syslog facility to be used which defaults to LOG\_USER.
- > `alert_email` : Email recipient address used by WebADM for sending system alerts. You can set several recipient addresses with the comma separator.
- > `cloud_service` : Enable RCDevs cloud service on your WebADM server. This is needed if you have a cloud license, want to benefit of Push login, Signature mechanisms, RCDevs SMS service, RCDevs cloud PKI...
- > `reverse_proxies` : If your WebADM server is used behind a reverse proxy or load-balancer, you need to set the IP address(es) of your reverse proxy server(s). Your proxy must be configured to create the HTTP\_X\_FORWARDED\_FOR and HTTP\_X\_FORWARDED\_HOST headers for WebADM to behave correctly. You should add your last reverse-proxy IP addresses separated by a comma to the `reverse_proxies` directive:

```
reverse_proxies "<YOUR_LASTREVERSEPROXY_IP1>", "<YOUR_LASTREVERSEPROXY_IP2>"
```

If you have more than one reverse-proxy between your WebADM server and clients, the `reverse_proxies` directive should be configured with number of reverse-proxies that are between the WebADM server and the clients, so WebADM is still able to get the actual client IP address. In that case, the directive must be configured like this:

```
reverse_proxies "<YOUR_LASTREVERSEPROXY_IP1> <NUMBER_REVERSE_PROXIES>", "  
<YOUR_LASTREVERSEPROXY_IP2> <NUMBER_REVERSE_PROXIES>"
```

- > `waproxy_proxies` : If you use WebADM Publishing Proxy from RCDevs (WAProxy) for publishing applications and services on public networks, then you must set the IP address(es) of the WAProxy server(s). Enable this setting ONLY if you are using RCDevs WAProxy as reverse-proxy! It is not intended to be used with any other reverse-proxies. You should add your WAProxy's IP addresses separated by a comma to the `waproxy_proxies` directive:

```
waproxy_proxies "<YOUR_WAPROXY_IP1>", "<YOUR_WAPROXY_IP2>"
```

If you have at least one or more reverse-proxy between your WAProxy server and clients, the `waproxy_proxies` directive should be configured with number of reverse-proxies (including WAProxy server) that are between the WebADM server and the clients, so WebADM is still able to get the actual client IP address. In that case, the directive must be configured like this:

```
waproxy_proxies "<YOUR_WAPROXY_IP> <NUMBER_REVERSE_PROXIES>", "<YOUR_WAPROXY_IP2> <NUMBER_REVERSE_PROXIES>"
```

- > `check_versions`: Enables WebADM versions checking. WebADM will check for new product versions for itself and for all the registered applications (web apps and web services).
- > `check_licenses`: Enables WebADM license update checking. WebADM will check if a license update is available on RCDevs online servers. This feature requires an Enterprise license to be already present.
- > `webapps_theme`: WebApps theme for WebApps. Only the default theme is available.
- > `unlock_message`, `unlock_subject`: Email message body and subject to be sent to a user when a WebApp access is temporarily unlocked by an administrator. The following variables are supported: %USERNAME%, %USERDN%, %USERID%, %DOMAIN%, %APPNAME%. These additional variables are available depending on the context: %APPNAME%, %APPID%, %TIMEOUT%, %EXPIRES%.
- > `org_name`, `org_logo`, `org_site`, `org_from`: You can customize your organization's name, logo file and website URL to be displayed in the Web applications. The logo file must be a PNG image with a size of 100x50 pixels, stored under the WebADM `conf/` directory. You can alternatively set the absolute path if the logo file is outside the WebADM config directory. The `org_from` allows you to configure the sender email address for emails sent by WebADM (ex. alerts, WebApp unlock...).
- > `treeview_items`: When an LDAP container which contains more than 1500 child objects is expanded in the Admin tree view, WebADM automatically displays an inline search input to filter the child results. The `treeview_items` defines the display limit and is set to 1500 by default.
- > `treeview_width`: This defines the default width (in pixel) for the tree view (left panel) in WebADM Admin Portal. In some circumstances, it can be useful to enlarge the tree view for a better display.
- > `default_portal`: It is possible to define which Portal corresponds to the default WebADM URL (without the trailing /admin, /webapps and /websrvs).

## 5.1 Other Configurations

You can create a `webadm.env` file in the WebADM `conf/` directory to modify some internal configurations such as port numbers and listen to an interface. You can change the following variables:

- > `INTERFACE`: Defines the network IP address to listen on. The default is 0.0.0.0 (any interface).
- > `HTTP_PORT_STD`: Defines the HTTP unsecured port used for the Admin Portal and WebApps. This port is not used and is a redirection to the `HTTP_PORT_SSL` port. The default port is 80.
- > `HTTP_PORT_SSL`: Defines the HTTP port over SSL used for the Admin Portal and WebApps. The default port is 443.
- > `SOAP_PORT_STD`: Defined the SOAP port in cleartext to be used for Web Services. The default port is 8080.
- > `SOAP_PORT_SSL`: Defined the SOAP port over SSL to be used for Web Services. The default port is 8443.
- > `CACHE_MEMSIZE`: Defines the memory size with a memory unit identifier to be allocated to the shared cache. The default size is 32 Mo (32M). This environment variable and the following ones are auto-adjusted by WebADM depending on the user scaling. You should not need to change it.

- > `REDIS_MEMSIZE`: Defines the memory size with a memory unit identifier to be allocated to the session manager. The default size is 256 Mo. Both `CACHE_MEMSIZE` and `REDIS_MEMSIZE` are auto-adjusted according to the number of users defined in the license file.
- > `REDIS_NOSYNC`: Defines if the WebADM session server replication should be disabled. This variable should be kept to its default value unless you really know what you are doing.

The format for the `webadm.env` file is:

```
INTERFACE=0.0.0.0
HTTP_PORT_STD=80
HTTP_PORT_SSL=443
SOAP_PORT_STD=8080
SOAP_PORT_SSL=8443
CACHE_MEMSIZE=32M
REDIS_MEMSIZE=256M
REDIS_NOSYNC=No
```

#### Note

WebADM will automatically adapt the threads and memory scaling according to the user amount (as defined in your Enterprise License). You generally do not need to touch these settings manually.

## 5.2 Encrypting Configuration Passwords

You can optionally encrypt any password in the configuration files for `webadm.conf`, `servers.xml` and `rsgnd.conf`. For example, you can use the tool `bin/pwcrypt` to convert a cleartext password to an encrypted form (ex. your WebADM Proxy User LDAP password). The encrypted password will look like `{wcrypt}ZuWw1le2qxlguTF77mDjmQ==`. You can use the new password value as is (ex. `proxy_password="{wcrypt}ZuWw1le2qxlguTF77mDjmQ=="`).

#### Note

This feature requires an Enterprise License if you're using WebADM version 1.7.\* or older. The encryption mechanism is bound to secret data in your encoded license file. The encryption is also per RCDevs customer and an encrypted password value cannot be used with another License.

## 6. LDAP Management

With WebADM, administrators can create and edit LDAP users, groups and other objects. Administrators can also extend existing LDAP users or groups with WebADM functionalities.

### 6.1 Common LDAP Objects

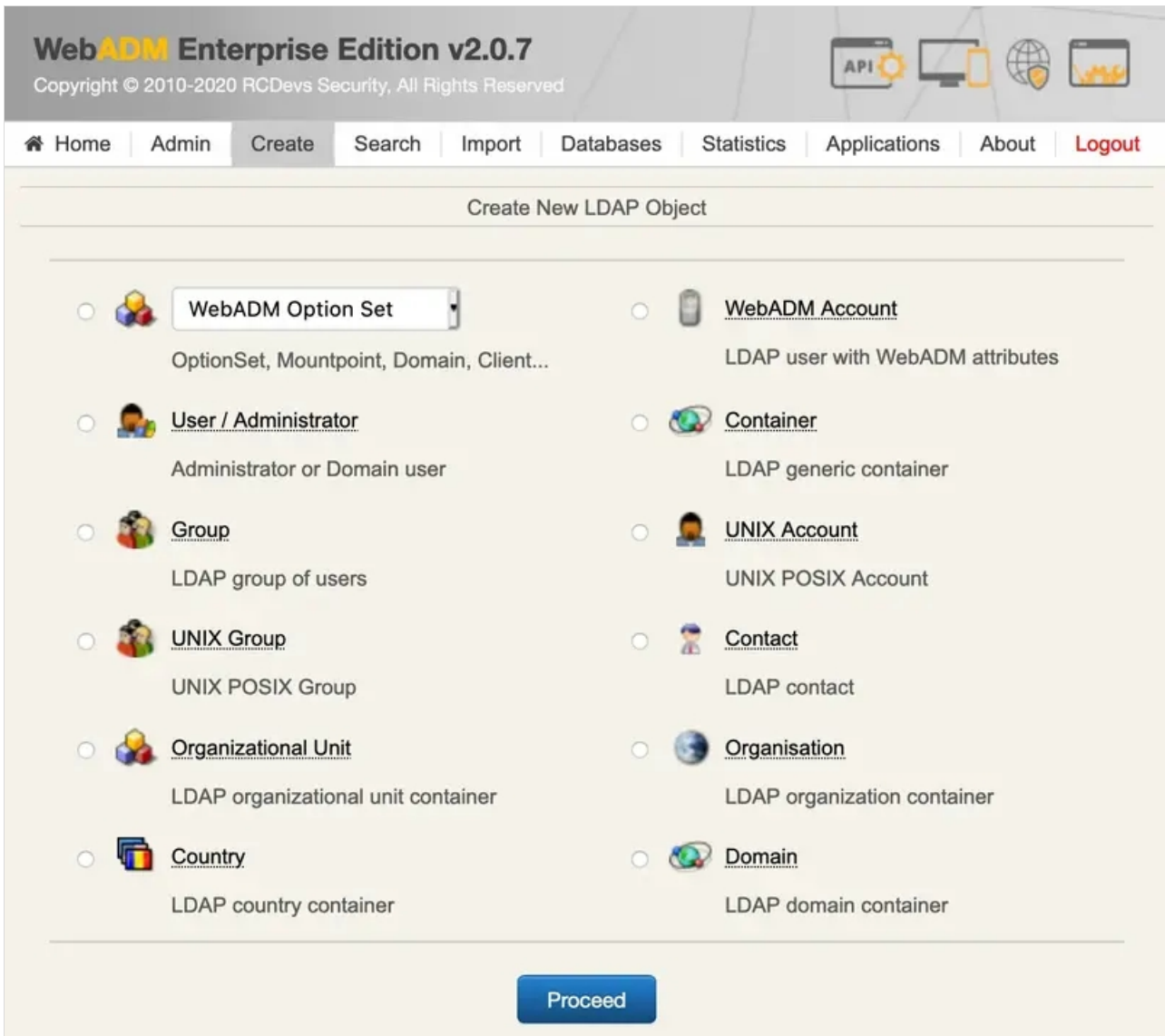


Figure 10a. Create New LDAP Objects

### 6.1.1 User Accounts

User accounts are LDAP standard user objects. WebADM considers a user account is an object containing at least one object class from the `user_oclasses` list in the WebADM main configuration file (`conf/webadm.conf`).

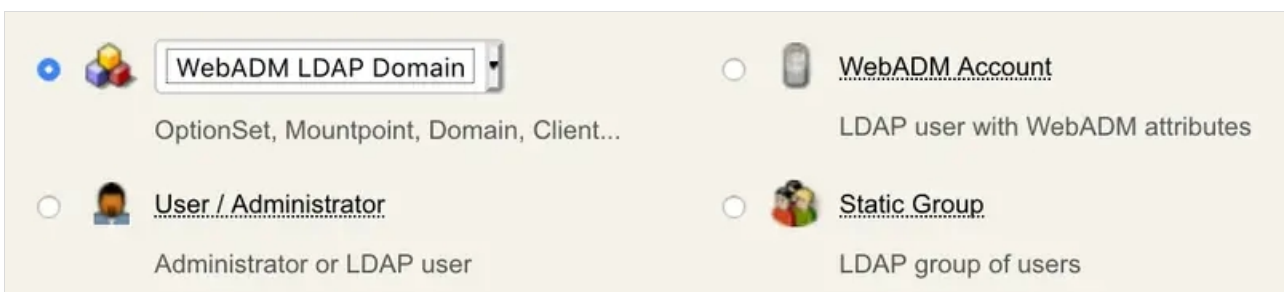


Figure 10. User Objects

WebADM provides its own user account schema named `webadmAccount`. The `webadmAccount` schema provides additional attributes, such as the `webadmSettings`, or `webadmData` for normal users and groups. These attributes are required by the registered Web Services and WebApps to store user-specific settings and user metadata. The `webadmAccount objectclass` is an

LDAP extension class and cannot be created standalone. It must be used together with a structural user object class. WebADM considers a WebADM account is an object containing at least one object class from the *webadm\_account\_oclasses* list in the WebADM main configuration file (`conf/webadm.conf`). In Figure 10, WebADM Account is an LDAP user object with the *webadmAccount* extension.

WebADM can create standalone users, or extend existing users by adding new object classes to the user object. See section Extending Objects for details.

The LDAP attribute corresponding to the login name (i.e. RADIUS username used for VPN logins) depends on the WebADM configurations. It can be the object name (CN) as well as the UID attribute, *sAMAccountName*, *userPrincipalName*, the mobile number, or anything else. WebADM just needs to know what attributes can be used for the logins. This is adjustable in the objects specification file (`conf/objects.xml`). By default, the username is the *UID* LDAP attribute.

WebADM accounts can contain several application settings. The list of available settings depends on the registered applications and the scope of the settings. Any Public or *LDAP* application setting can be set at the user or group level.

WebADM and its applications use LDAP bind for static password checking. That means that the user objects must be combined with a bindable object class and must have their LDAP password set.

### ⚠ Important

To be used by the Web Services and WebApps, an LDAP user **must** be a WebADM account. WebADM user accounts are those containing the *webadmAccount* LDAP object class. You can enable the WebADM features on any existing LDAP user by extending it with the *webadmAccount* extension (with the object extension action in the object editor).

You can assign WebADM settings to LDAP groups (instead of users) by extending the groups with the *webadmGroup* extension. Like with users, this is done with the object extension action in the object editor.

### 6.1.2 User Groups

User groups are LDAP object that contains a list of LDAP members, each representing the Distinguished Name (DN) of the user object belonging to the group. WebADM considers a group is an object containing at least one object class from the *group\_oclasses* list in the WebADM main configuration file (`conf/webadm.conf`).

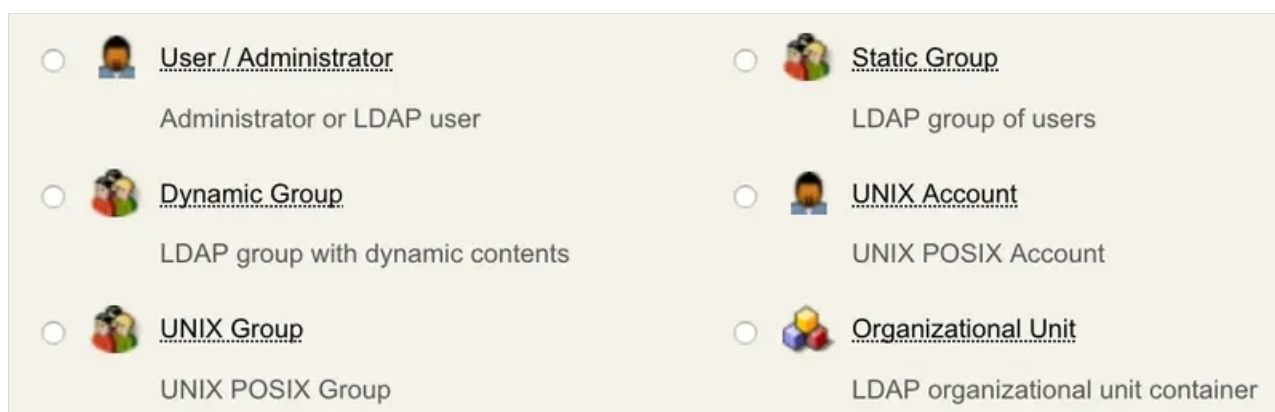
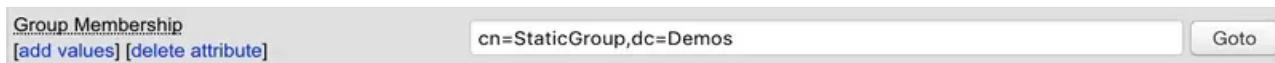


Figure 11. Group Objects

Groups are often used for access purposes. Members of a particular group are allowed to access different services than members of another. Groups are also used for storing application settings common to all group members. This often reduces the overhead in managing settings stored in user accounts. In that case, the groups must be extended with the `webadmGroup` object class.

WebADM supports two methods to assign users to groups:

1. Using group membership: The user account includes a *groupmembership* (such as *memberOf*) attribute specifying which group DNs it belongs to.



A screenshot of a web form titled "Group Membership". On the left, there are links for "[add values]" and "[delete attribute]". In the center, there is a text input field containing the value "cn=StaticGroup,dc=Demos". On the right, there is a "Goto" button.

Figure 12. Group Membership User Attribute

2. Using group members: The group object includes a member list containing the user DNs.

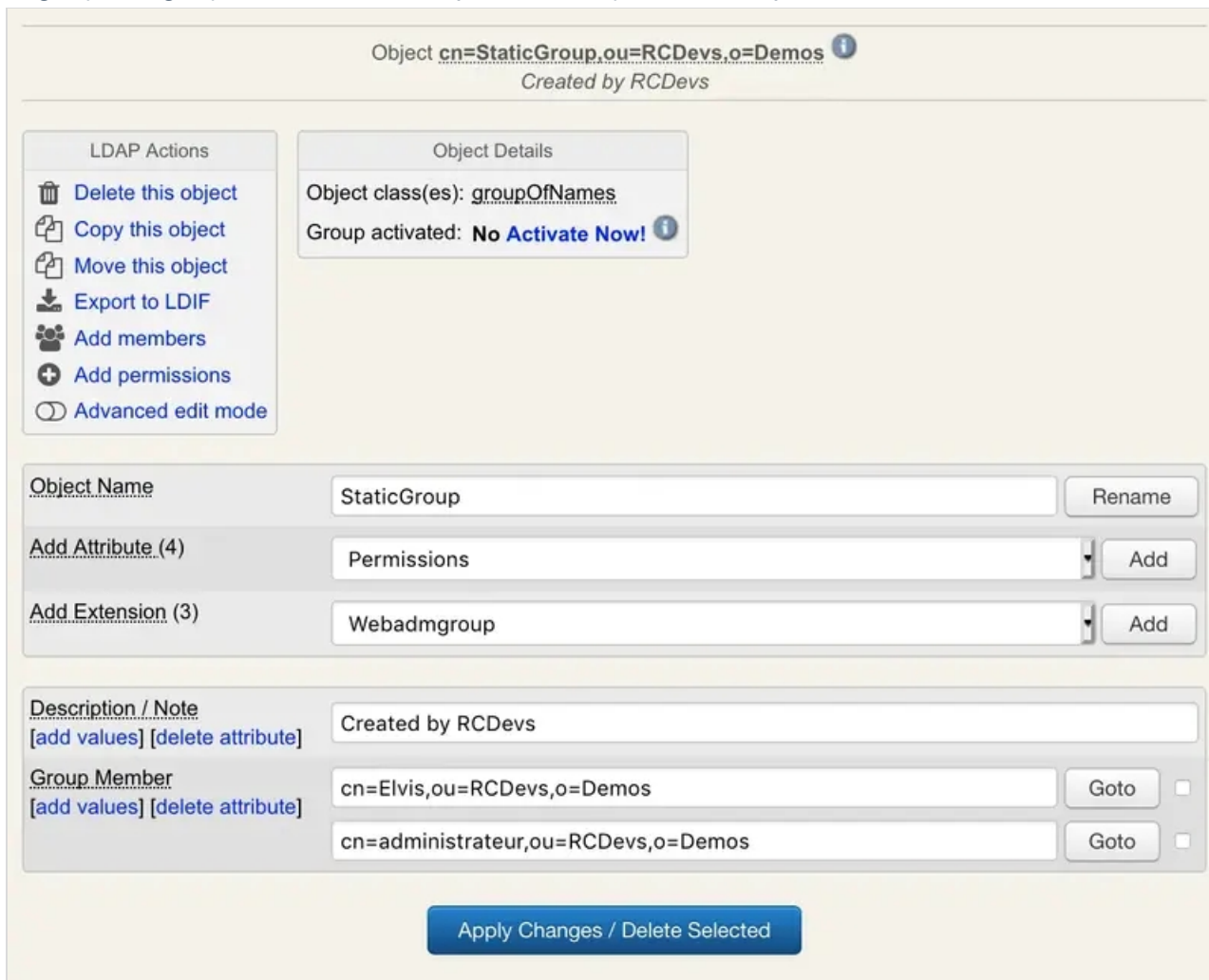


A screenshot of a web form titled "Group Member". On the left, there are links for "[add values]" and "[delete attribute]". In the center, there is a text input field containing the value "cn=TestUser,dc=Demos". On the right, there is a "Goto" button.

Figure 13. Group Member Attribute

WebADM provides two ways to define groups:

1. Static groups. The group member list is statically defined and updated manually.



A screenshot of a web page for configuring a static group. At the top, it shows the object path "Object cn=StaticGroup,ou=RCDevs,o=Demos" and "Created by RCDevs". On the left is a "LDAP Actions" menu with options like "Delete this object", "Copy this object", "Move this object", "Export to LDIF", "Add members", "Add permissions", and "Advanced edit mode". In the center is an "Object Details" box showing "Object class(es): groupOfNames" and "Group activated: No Activate Now!". Below this are three rows for configuration: "Object Name" (StaticGroup), "Add Attribute (4)" (Permissions), and "Add Extension (3)" (Webadmgroup). At the bottom, there is a "Description / Note" field with "Created by RCDevs" and a "Group Member" list containing "cn=Elvis,ou=RCDevs,o=Demos" and "cn=administrateur,ou=RCDevs,o=Demos". A large blue button at the bottom says "Apply Changes / Delete Selected".

Figure 14. Static Group



- Dynamic groups. The group member list is defined as a dynamic LDAP query (*Dynamic Member Query*). The member list is computed at runtime when the group members are queried. Dynamic groups require a *Dynamic Group Query Identity attribute* which defines the user account DN to be used internally by Novell eDirectory, for performing the LDAP searches needed for looking up the dynamic group members.

You can assign WebADM settings to LDAP groups (instead of users) by extending the groups with the *webadmGroup* extension. Like with users, this is done with the object extension action in the object editor.

## Note about group settings

User groups and group settings are cached for 5 minutes in order to optimize group searches and user setting resolutions. This has the side effect that user groups and group settings' changes may be delayed for a maximum time of 5 minutes when used by WebApps and Web Services.

The screenshot shows the 'Object Editor' for a dynamic group. The object is identified as 'cn=DynamicGroup,ou=RCDevs,o=Demos' and was created by 'RCDevs'. The interface is divided into several sections:

- LDAP Actions:** A sidebar menu with options: Delete this object, Copy this object, Move this object, Export to LDIF, Change password, Add members, Add permissions, and Advanced edit mode.
- Object Details:** Shows 'Object class(es): dynamicGroup, groupOfName...' and 'Group activated: No Activate Now!'.
- Object Name:** 'DynamicGroup' with a 'Rename' button.
- Add Attribute (7):** A dropdown menu showing 'Permissions' and an 'Add' button.
- Add Extension (2):** A dropdown menu showing 'UNIX Group' and an 'Add' button.
- Description / Note:** 'Created by RCDevs' with '[add values]' and '[delete attribute]' links.
- Dynamic Group Query Identity:** 'cn=admin,o=Root' with a 'Select' button and a '[delete attribute]' link.
- Group Member:** A list of members: 'cn=Elvis,ou=RCDevs,o=Demos', 'cn=Philippe,ou=RCDevs,o=Demos', and 'ou=App Access,ou=RCDevs,o=Demos'. Each entry has a 'Goto' button and a checkbox.
- Dynamic Member Query URL:** 'ldap:///OU=App%20Access,OU=RCDevs,O=Demos??base?(objectclass=dynamicGroup)' with 'List' and 'Edit' buttons.

At the bottom, there is a large blue button labeled 'Apply Changes / Delete Selected'.

Figure 15. Dynamic Group

WebADM provides administration pages to define dynamic queries and check their content.

## Note

Dynamic groups are supported on Novell eDirectory and RCDevs Directory Server only.

### 6.1.3 Administrative Accounts

WebADM considers that any bindable LDAP user object with sufficient access rights to the LDAP server can be used for LDAP administration purposes, with access to the Admin Portal or the Manager interface. Access to any of these WebADM management interfaces requires the LDAP users to be configured in the *super\_admins* list in `conf/webadm.conf` or to be part of a WebADM AdminRole (see the AdminRoles section for details).

In WebADM, administrators can create and edit LDAP objects, create new contexts with sub-administrators, and assign permissions in the boundaries of their own LDAP restrictions and permissions. It is possible to restrict which features and operations delegated administrators can access by using WebADM OptionSets and AdminRoles.

### 6.1.4 Permissions

By default, a newly created administrator has no write permissions.

With Novell eDirectory, write permissions must be created by adding permission attributes (ACL) to LDAP contexts. You can use the Add Permissions action when editing a container object to create new permissions.

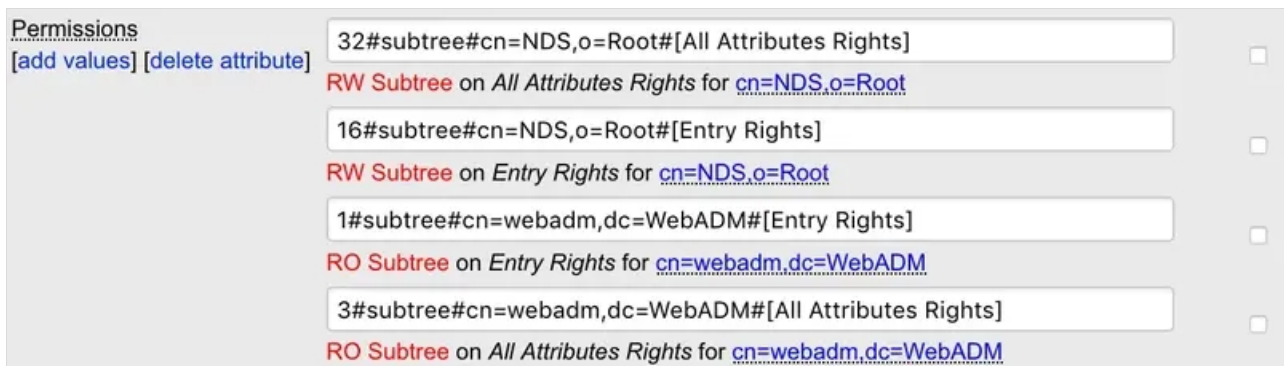


Figure 16. LDAP Permissions (ACL) Attribute

- › With Microsoft ActiveDirectory, the user must be added to an administrative group where fine-grained permissions configured on that group according to what you want to allow in terms of LDAP manipulation.
- › With OpenLDAP, the user permissions must be added in the OpenLDAP server configuration file (`slapd.conf`).

By default, a user does not have any other rights than reading access. He is able to manage his own user data, change his password or renew his own certificates. If another administrator creates context permissions for him, he becomes an administrator in these contexts.

### 6.1.5 Certificates

WebADM supports certificate-based authentication for simpler and more secure access to the Administration Portal and

WebApps. It provides the necessary pages and actions to easily manage administrator certificates. Supported operations are certificate creation, deletion, renewal, download.

Certificate-based authentication is highly recommended when using delegated administration and especially when using WebADM for remote administration over the Internet. It adds another level of security while authorizing the administrator's sessions at the web server's level. WebADM provides a wizard to create a new administrator certificate and parameters allow to set the type and validity time for the new certificates. See the Managing Certificates section for details.

An administrator is able to renew, remove or add new certificates for other users he manages or for himself.



Figure 17. User Certificate Attribute

The WebADM internal PKI (RSignd) is the default certificate management system. If you already have an internal PKI, you can configure WebADM as a subordinate certificate authority. Refer to the following [documentation](#) to configure it.

## Note

Any bindable object is able to log into WebADM (if declared as super/other admins). That means any user object with a bind password and a login certificate is able to enter WebADM. To prevent normal users from logging in WebADM, use the certificate-based login instead of the Domain or DN login modes. Then, only administrators owning a valid administrator certificate can log in.

## 6.1.6 Containers

LDAP containers (or contexts) are objects, which can contain child objects. WebADM considers a container is an object containing at least one object class from the *container\_oclasses* list in the WebADM main configuration file (`conf/webadm.conf`). Common containers are *Organizations*, *Organizational Units*, *Countries*, *Locations*, *Domains*, etc... When WebADM is used to manage a lot of users, it is highly recommended to structure the LDAP tree with containers in order to reduce the number of child objects within one container.

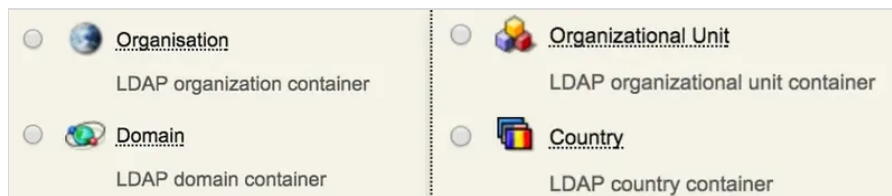


Figure 18. Container Objects

An administrator can assign LDAP permissions and OptionSets on containers.

## 6.2 WebADM Configuration Objects

Configuration objects are WebADM-specific LDAP objects that are used by WebADM for storing persistent configurations in LDAP. WebADM considers an LDAP configuration object is an object containing at least one object class from the

webadm\_config\_oclasses list in the WebADM main configuration file (`conf/webadm.conf`). The type of the configuration object is determined by the webadmConfig attribute which can be either *Domains*, *Clients*, *AdminRoles*, *OptionSets*, *MountPoints*, *WebApps* or *WebSrvs*.

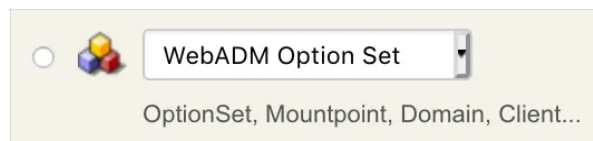


Figure 19. Configuration Objects

WebADM creates a set of LDAP containers and objects during its setup for storing WebADM *Domains*, *OptionSets*, *MountPoints*, *client policies*, *Web Services* and *WebApps* configurations. The LDAP locations for these objects are defined in the main WebADM configuration file (`conf/webadm.conf`). This tree structure is mandatory for WebADM to operate correctly.



Figure 20. WebADM Tree Structure

WebADM configuration objects are accessible from the Admin menu or directly from the dc=WebADM subtree (can also be an OrganizationalUnit (OU), AD container (CN)..)

### 6.2.1 WebADM AdminRoles

WebADM includes the concept of delegated administration. It also makes the distinction between Super Administrators and Other Administrators. Super Administrator is an LDAP administrator (ex. AD Domain Admin users) which are configured in the *super\_admins* list in `conf/webadm.conf`. The Super Administrators have unlimited access to any feature of WebADM. On the contrary, Other Administrators are the delegated administrators for which you can define precisely what features and administration operations are allowed through WebADM AdminRole objects. Another Administrator is also any LDAP user which is a member of one or several WebADM AdminRole(s).

LDAP access rights for both Super Administrators and Other Administrators MUST be set at the LDAP server level with dedicated LDAP ACLs. Any action through WebADM admin GUI or Manager APIs are performed with the permissions of the authenticated user and these permission will be needed on the LDAP in order complete needed operations. This is important to notice that WebADM enforces access control over its own management interfaces but it cannot enforce any security control at the LDAP API level! This means that restricting user operations and features via AdminRole configurations does not prevent an administrator from performing the same operations from another LDAP client software.

All AdminRoles must be stored in the same container (as specified in the WebADM main configuration file) to be read by WebADM at start-up.

### 6.2.2 WebADM OptionSets

Some WebADM restrictions or “subtree options” can be assigned to specific LDAP contexts using WebADM OptionSets. OptionSets are essentially subtree profiles which can be used for example to define a unicity verification context or limiting the LDAP view depth for delegated administrators. Option sets can also be used to create Organization profiles specifying the default LDAP attributes for member objects within the organization. See section *WebADM OptionSets* for details.

All OptionSets must be stored in the same container (as specified in the WebADM main configuration file) to be read by WebADM at start-up.

### 6.2.3 WebADM MountPoints

MountPoints are containers in the LDAP tree that include objects and child containers (i.e. the entire tree structure) from another LDAP server. The objects are not physically present in the tree structure. Instead, WebADM connects at runtime to an external LDAP and renders its contents as if the data were stored in the mounted context. See section *WebADM MountPoints* for details.

All MountPoints objects must be stored in the same container (as specified in the WebADM main configuration file) to be read by WebADM at start-up.

### 6.2.4 WebADM LDAP Domains

All the WebADM applications identify a user with a username, a password and a domain name. The domain objects establish the relationship between a domain name and an LDAP tree base. Also, when an application wants to obtain an LDAP user DN corresponding to the provided login information, it will use the domain tree base to build the LDAP search. See section *WebADM Domains* for details.

All Domains objects must be stored in the same container (as specified in the WebADM main configuration file) to be read by WebADM at start-up.

### 6.2.5 WebADM Client Policies

A Client Policy provides per-client application access control and customized configurations. The Client Policy objects are also used to customize the behavior of a client application (ex. a VPN server using OpenOTP Authentication Server).

You can create a client policy object having the name of a Web Service’s client ID. For example, you use the client names as displayed in the WebADM log viewer for the client object names.

When a client is defined, any request from the corresponding client application (ex. a VPN server with matching client ID), will obey the defined client policy.

For a client, you can restrict users able to use the client application with allowed and excluded group lists. And you can define some Web Service settings which will always be enforced for the client. For example, you want the VPN to authenticate users with LDAP+OTP passwords and Token, whatever policy is defined for the user.

### 6.2.6 WebADM WebApps and Web Services

These objects are used by WebADM to store registered application configurations. When you register a new application in WebADM, it creates an LDAP object. You can access the application configuration either by editing the application LDAP object or using the Applications menu in WebADM.

All WebApps and WebServices configuration objects must be stored in the same container (as specified in the WebADM main configuration file) to be read by WebADM at start-up.

## 6.3 WebADM-Specific Attributes

WebADM schema (see section *WebADM LDAP Schema* for details) provides two additional object classes : *webadmAccount* and *webadmConfig*.

LDAP objects extended or created with the *webadmAccount* and *webadmGroup* object class support the following new attributes.

### 6.3.1 WebADM Settings Attribute

This attribute is used to store user-specific application settings inside the user or group objects. The object settings have priority over the default application settings for the registered WebADM applications.



Figure 21. *webadmSetting* Attribute

The WebADM user setting editor displays a drop-down list containing the registered applications. Select an application and the list of corresponding settings are displayed and available for configuration.

### 6.3.2 WebADM Data Attribute

This attribute is used by the WebADM applications to store user data such as Token keys and various user data. The WebADM user data editor displays all the data stored by the applications and allows raw edition of the data.

WebADM User Data [delete attribute]	Edit Application Data
<a href="#">OpenOTP.Device1Data:</a>	[BINARY DATA - 129 Bytes]
<a href="#">OpenOTP.Device1Name:</a>	Yubico U2F EE Serial 13503277888
<a href="#">OpenOTP.Device1State:</a>	0
<a href="#">OpenOTP.Device1Type:</a>	FIDO2
<a href="#">OpenOTP.LastLogin:</a>	2018-05-14 11:49:52
<a href="#">OpenOTP.LoginCount:</a>	1
<a href="#">OpenOTP.RejectCount:</a>	5
<a href="#">OpenOTP.TokenID:</a>	IOS:b0e5792c770a287a277ad9afaecd58f5d5e9...
<a href="#">OpenOTP.TokenKey:</a>	[PROTECTED BINARY DATA - 20 Bytes]
<a href="#">OpenOTP.TokenModel:</a>	iPhone10,5
<a href="#">OpenOTP.TokenSerial:</a>	39DEE717-500D-4B31-BF90-A845FC7D81A7
<a href="#">OpenOTP.TokenState:</a>	0
<a href="#">OpenOTP.TokenType:</a>	TOTP

Figure 22. webadmData Attribute

The webadmData contents are encrypted in the LDAP using an AES-256 key which is configured in the WebADM main configuration file ( `conf/webadm.conf` ). Only the WebADM administrators are able to read this attribute unencrypted. They can edit the data values with the data editor.

### ⚠ Important note for Advanced encryption

The webadmData encryption uses the LDAP DN together with the encryption key. This is a security mechanism to prevent the same data values from two different users to be encrypted identically. When a user is copied, WebADM handles the re-encryption automatically. But if you export the user in an LDIF file, and re-import it at another location, the webadmData are lost. Features are automatically disabled.

### 6.3.3 Password Attribute

Any bindable LDAP object must have its password attribute set. Password attributes are defined by the password\_attrs setting in the WebADM main configuration file ( `conf/webadm.conf` ). The password encoding and format depends on the LDAP directory type. The encoding and encryption format of passwords is defined in the objects specification file ( `conf/objects.xml` ).

Administrators can change user passwords with the *Change password* action in the object editor.

The screenshot shows the WebADM Freeware Edition v2.0.7 interface. On the left, there is a tree view of the LDAP directory structure under 'LDAP\_Server1 (RCDevs Directory)'. The tree includes 'dc=WebADM', 'o=Demos', and 'o=Root (7)'. Under 'o=Root', several users are listed: 'cn=admin', 'cn=admins\_group', 'cn=marcus', 'cn=ppolicy', 'cn=user1', and 'cn=user2'. The main panel displays the 'Change Password for cn=admin, o=Root' form. It includes a 'New Password:' field, a 'Confirm Password:' field, and two buttons: 'Update Password' and 'Cancel'. The top navigation bar contains links for Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout.

Figure 23. Changing User Password

### 6.3.4 UID Attribute

WebADM allows specifying multiple attributes to be used as login attributes. Login attributes are defined by the `uid_attrs` setting in the WebADM main configuration file (`conf/webadm.conf`).

When a user logs in a WebApp or a Web Service, he enters his login name, domain and password. WebADM computes the LDAP tree base using the information stored in the Domain configuration object and searches for objects of type `webadm_account_ouclasses`, having one `uid_attrs` corresponding to the provided login name. Then WebADM binds the LDAP directory with the user DN and the provided password.

The same system is used when WebADM Administrator Portal is configured in Domain login mode. But in that case, WebADM will search for any user object of type `user_ouclasses`, having one `uid_attrs` corresponding to the provided login name.

### 6.3.5 Certificate Attribute

WebADM uses this attribute to store user certificates in the LDAP accounts.

#### Note

The private keys are never stored in this attribute. Certificate attributes are defined by the `certificate_attrs` setting in the WebADM main configuration file (`conf/webadm.conf`).

WebADM supports storing user certificates in both binary and base64 encoding. The encoding is specified in the objects specification file (`conf/objects.xml`).

### 6.3.6 Language Attribute

This attribute is used by the WebADM applications to query the user language. When application messages are localized in several languages (with the WebADM Localized Message Editor), the applications will automatically select the message corresponding to the user language.

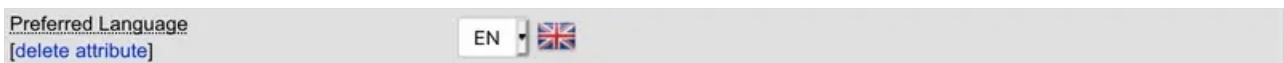


Figure 24. preferredLanguage Attribute

### 6.3.7 Mobile Attribute

This attributes stores the user mobile phone number. It is used by some WebADM applications and services.

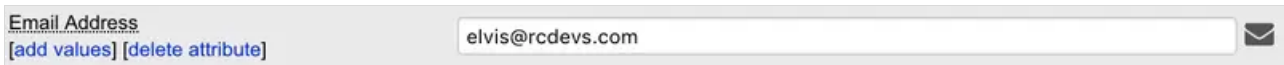


Figure 25. mobile Attribute



### 6.3.8 Mail Attribute

This attributes stores the user email address. It is used by some WebADM applications.



Mail Attribute configuration interface showing the value `elvis@rcdevs.com` and control links `[add values]` and `[delete attribute]`.

Figure 26. mail Attribute

### 6.3.9 Voice Attribute

This attribute stores the user biometric voice. It is required when OTP Type setting is configure to VOICE in OpenOTP.

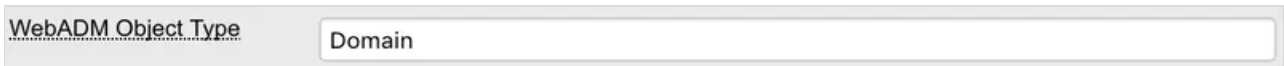


WebADM Voice Model configuration interface showing the value `[BIOMETRIC VOICE MODEL - 196 KBytes]` and control links `[add values]` and `[delete attribute]`.

Figure Voice. Voice Attribute

### 6.3.10 WebADM Config Type Attribute

This attribute is used by WebADM to assign a role to a webadmConfig object (examples of WebADM types are *Domain*, *Trust*, *OptionSet*, *MountPoint*, *Client*, *WebApp* or *WebSrv*).



WebADM Object Type configuration interface showing the value `Domain` and control link `[add values]`.

Figure 27. webadmType Attribute

## 6.4 WebADM LDAP Schema

The WebADM LDAP schema extension provides 3 additional object classes: *webadmAccount*, *webadmGroup* and *webadmConfig*. See the figure below for the schema detail.

Objectclass webadmAccount		
OID 1.3.6.1.4.1.34617.2.4.1 - Type Auxiliary		
Required Attributes	Optional Attributes	Superiors
<ul style="list-style-type: none"> <li>• <a href="#">cn</a></li> <li>• <a href="#">sAMAccountName</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">description</a></li> <li>• <a href="#">mail</a></li> <li>• <a href="#">mobile</a></li> <li>• <a href="#">preferredLanguage</a></li> <li>• <a href="#">webadmVoice</a></li> <li>• <a href="#">webadmData</a></li> <li>• <a href="#">webadmSettings</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">top</a></li> </ul>
Objectclass webadmConfig		
OID 1.3.6.1.4.1.34617.2.4.2 - Type Structural		
Required Attributes	Optional Attributes	Superiors
<ul style="list-style-type: none"> <li>• <a href="#">cn</a></li> <li>• <a href="#">webadmType</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">description</a></li> <li>• <a href="#">webadmSettings</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">top</a></li> </ul>
Objectclass webadmGroup		
OID 1.3.6.1.4.1.34617.2.4.3 - Type Auxiliary		
Required Attributes	Optional Attributes	Superiors
<ul style="list-style-type: none"> <li>• <a href="#">cn</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">description</a></li> <li>• <a href="#">webadmSettings</a></li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">top</a></li> </ul>

Figure 28. WebADM Schema

The new LDAP schema entries are automatically registered in the LDAP server schema by the WebADM setup.

## 6.5 Creating Objects

With WebADM, you can create any object type defined in the objects specification file (`conf/objects.xml`). Yet, if the object is not present in the LDAP schema, it is ignored. The objects specification file defines additional information used by WebADM about the object types and their capabilities. It defines what administrative level is required by an administrator to create an object, the additional object classes to be merged during creation and the available extension classes.

Administrative levels are used to set up level-based object creation restrictions. They are used to control what objects can be created by administrators belonging to a given context. An object specification also includes the minimum administrative level required for the object to be created.

The *auxclasses* in the object specification is used to consider a set of object classes as a single WebADM object type. It is mandatory for certain object types such as WebADM accounts because the *webadmAccount* object class is an LDAP extension class. That means, it cannot create a standalone and must be associated with a structural objectclass which defines other LDAP attributes.

The extensions define the object classes with which the object can be extended. For example, an object class corresponding to existing user objects should be extendable with the *webadmAccount* object class to allow adding WebADM features and settings to existing users.

Objects can be created either from the top menu Create button or directly from the Create button within a context in the LDAP tree. The creation forms will depend on the OptionSets applying on the creation context and on the LDAP schemas corresponding to the new object DN in case of MountPoint.

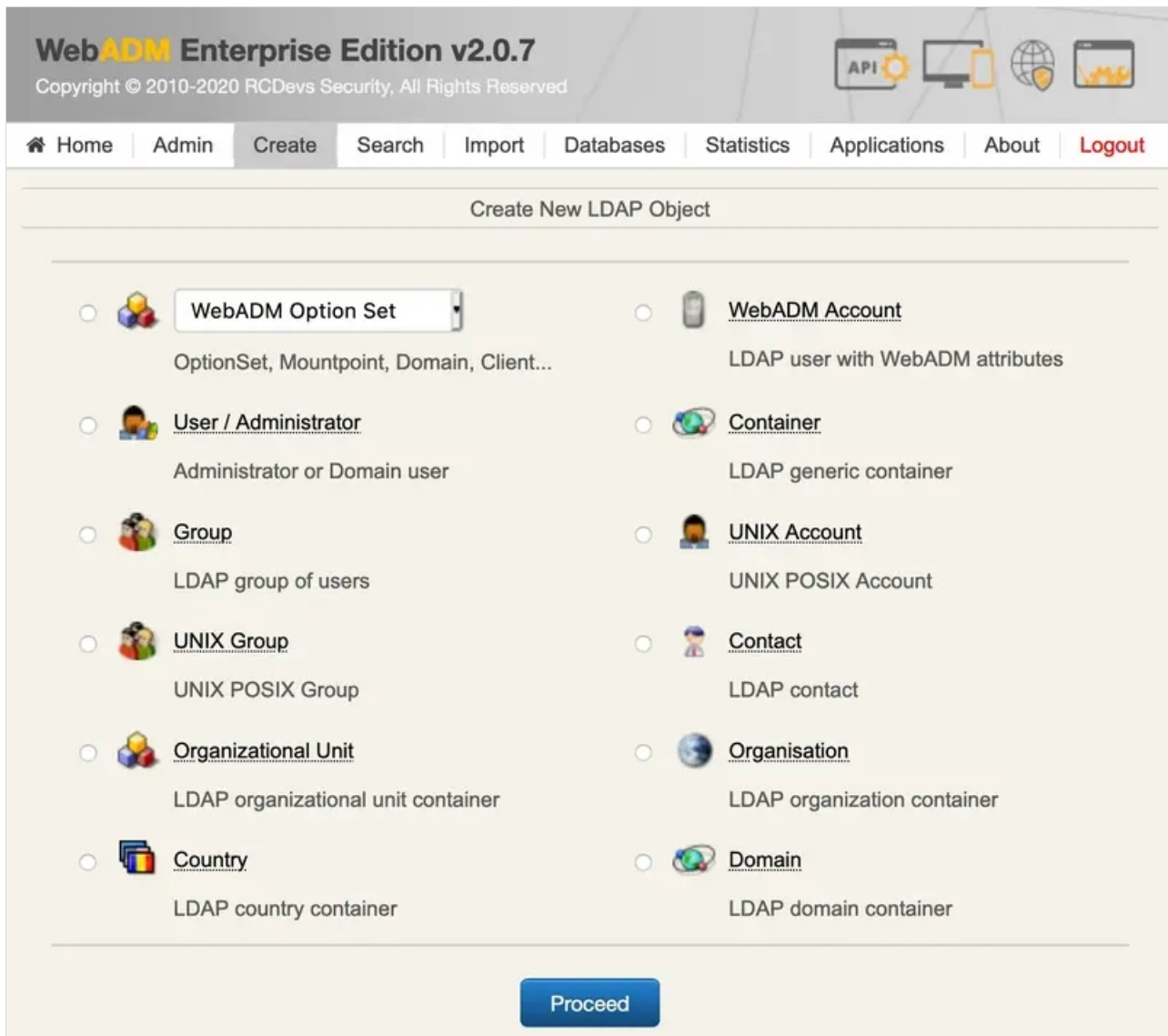


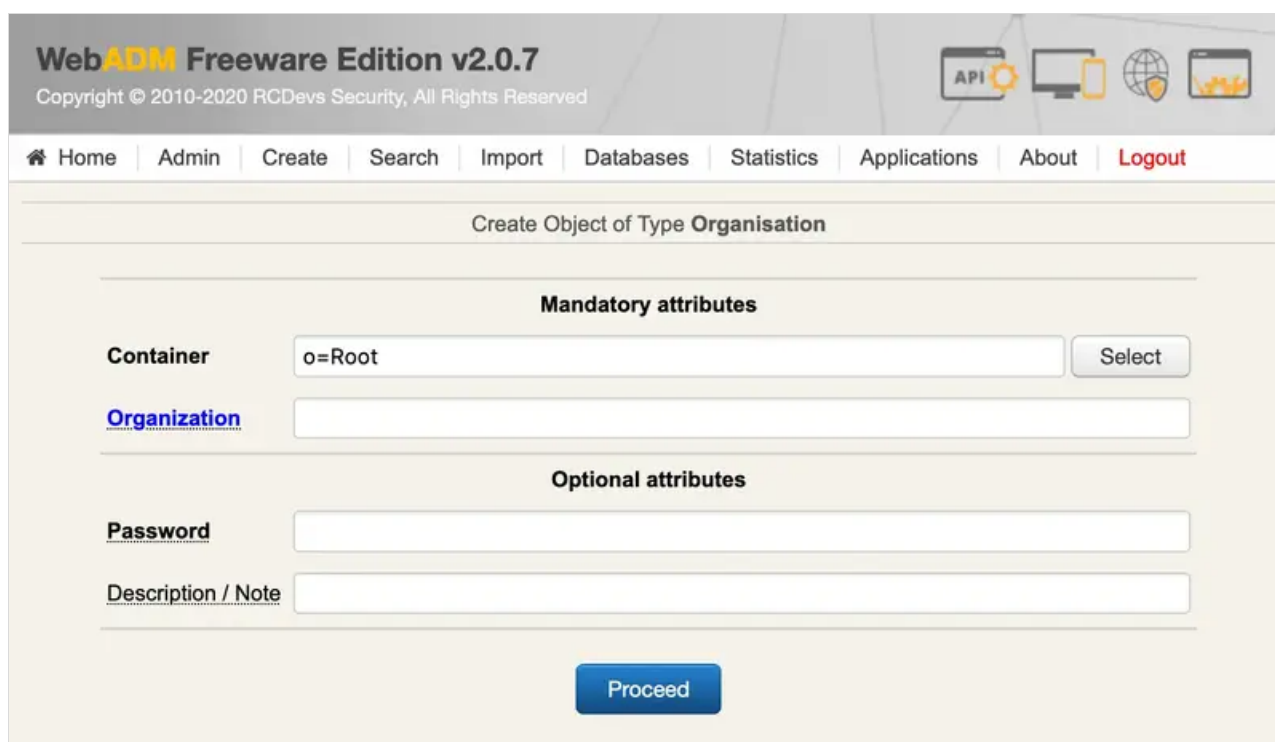
Figure 29. Create Object List

The object creation forms are computed dynamically by querying the LDAP schema for the object class and *auxclasses*, associated attributes and the constraints. They display all the mandatory attributes (merged from all objectclasses) and the optional attributes to be created.

## Note

Only those optional attributes configured in the attribute specifications of the WebADM objects specification file (conf/objects.xml) are displayed. This is a display simplification not to show all the merged optional attribute list which can be very long depending on your LDAP.

Some attribute values can be autofilled if default values are defined in the *OptionSets* which apply on the object creation context.



The screenshot shows the 'Create Object of Type Organisation' form in the WebADM Freeware Edition v2.0.7 interface. The form is titled 'Create Object of Type Organisation' and is divided into two main sections: 'Mandatory attributes' and 'Optional attributes'. In the 'Mandatory attributes' section, there is a 'Container' field with the value 'o=Root' and a 'Select' button, and an 'Organization' field. In the 'Optional attributes' section, there is a 'Password' field and a 'Description / Note' field. A 'Proceed' button is located at the bottom of the form. The interface also shows a navigation menu with items like Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout.

Figure 30. Create Object Form

The creation wizard includes a WebADM Config Object item (with a drop-down list) in the new objects list. This kind of object corresponds to WebADM configuration objects such as *Domains*, *MountPoints*, *OptionSets*, *WebApps* or *WebSrvs*.

## 6.6 Editing Objects

The object editor displays useful information about the object, a list of actions to be performed and the list of attributes contained by the object.

### 6.6.1 The Contextual Action Box

The action box is displayed at the top left of the editorial page. It contains a list of actions to be performed on the object such as deletion, copy, LDIF export, child creation, add permissions, issue certificate... It includes a button to change the user password if the object is used for LDAP binds.

When the edited object is a container containing child objects, copy, delete and export operation can be performed recursively.

A button allows switching to advanced edition mode. By default, the edition form does not display all the object attributes nor all

the edition capabilities or attribute list. It displays all the mandatory attributes but only the optional attributes which are defined in the objects specification file (`conf/objects.xml`). And the behavior is the same for extension classes list and new attributes list. If required, you can at any moment switch to advanced mode for extended display and capabilities.

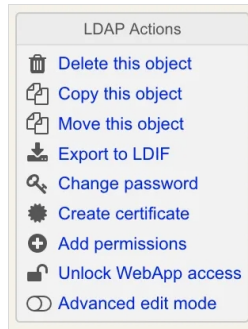


Figure 31. Action Box

### 6.6.2 The Information Box

The object informational box is displayed at the top middle of the editorial page. It displays useful information for the object such as a unicity check, WebADM settings, data summary, etc...

If some attributes are defined as unique within a specific context, WebADM checks the unicity and display the result and the list of checked attributes in this box. If attributes have to be unique, this must be set in the objects specification file (`conf/objects.xml`).



Figure 32. Information Box

### 6.6.3 The Application Box

This box is displayed at the top right of the edition page (only when an application is registered). All the registered applications can specify some additional actions to be performed by WebADM administrators as part of the user management. Those actions are generally accessible in this box (for the administrators) and through the SelfDesk WebApp (for the end-users).



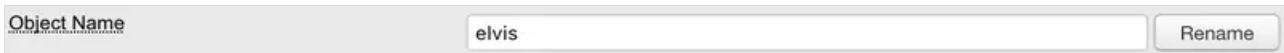
Figure 33. Application Box

### 6.6.4 Object Name

The object name is the value of the LDAP naming attribute for the object. You can change the object name by typing a new name and using the rename button. Generally, the naming attribute is the object Common Name (CN).

#### Note

You cannot rename a container object which already contains child objects. But you can recursively copy the container to a new one (with another name).




The screenshot shows a dialog box titled "Object Name". On the left, the text "Object Name" is displayed. To its right is a text input field containing the word "elvis". Further to the right is a button labeled "Rename".

Figure 34. Rename Object

### 6.6.5 New Attributes

The Add Attribute button allows adding optional attributes supported by any of the object classes composing the object.

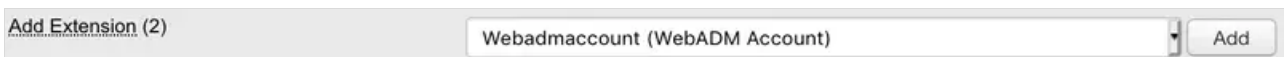


The screenshot shows a dialog box titled "Add Attribute (48)". On the left, the text "Add Attribute (48)" is displayed. To its right is a dropdown menu with "Postaladdress" selected. Further to the right is a button labeled "Add".

Figure 35. Add Attribute

### 6.6.6 Extensions

The Add Extension button allows adding new compatible object classes to the object. When adding an extension, a wizard will ask for the new mandatory attributes and the optional attributes which are defined in the objects specification file (`conf/objects.xml`).



The screenshot shows a dialog box titled "Add Extension (2)". On the left, the text "Add Extension (2)" is displayed. To its right is a dropdown menu with "Webadmacro (WebADM Account)" selected. Further to the right is a button labeled "Add".

Figure 36. Add Extensions

To see the list of object classes in an object switch to advanced edition mode.

You can remove an extension object class from an LDAP object by switching to advanced edition mode, checking the object class checkbox (in the object class attribute list), and clicking the *Apply Changes / Delete Selected* button. The object class removal will also remove the object class and all the attributes that are not part of any of the remaining object classes.

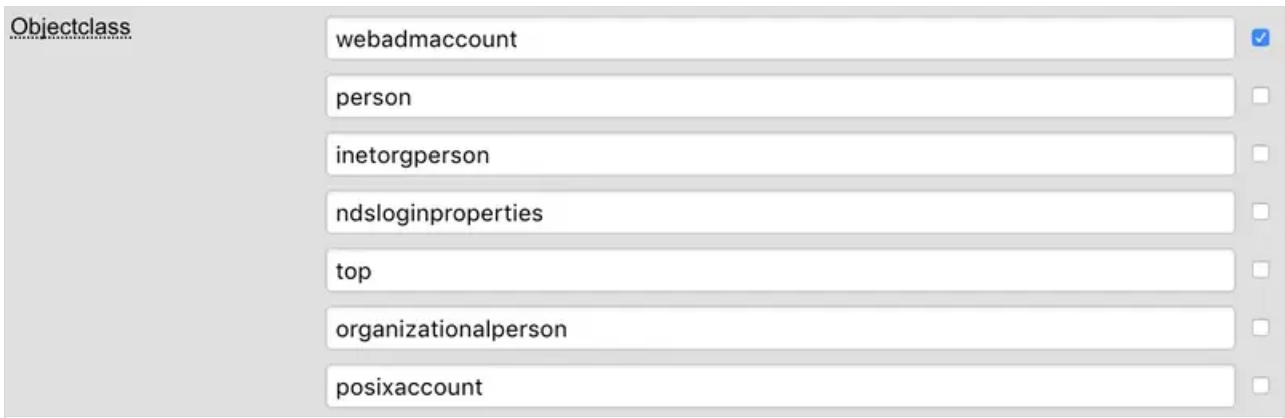


Figure 37. Remove Objectclass

### 6.6.7 Attribute List

The attribute list displays the attributes which have a value defined in the object.

#### Note

Only the attributes defined in the objects specification file (conf/objects.xml) are displayed by default. This is a display simplification to ease the use of WebADM but you can display all the attributes by switching to the advanced edition mode.


<u>Description / Note</u> [add values] [delete attribute]	Created by RCDevs	
<u>Logintime</u> [delete attribute]	20181026131726Z	
<u>Objectclass</u>	webadmaccount <input type="checkbox"/>	
	person <input type="checkbox"/>	
	inetorgperson <input type="checkbox"/>	
	ndsloginproperties <input type="checkbox"/>	
	top <input type="checkbox"/>	
	organizationalperson <input type="checkbox"/>	
<u>Preferred Language</u> [delete attribute]	EN 	
<u>Last Name</u> [add values]	Testing	
<u>Login Name</u> [add values]	Testing	
<u>WebADM User Data</u> [delete attribute]	OpenOTP.TokenType={wcrypt}IRkykeuKsEs4ViLCEuFvkw==,OpenOTP.TokenID: IOS:b0e5792c770a287a277ad9afaecd58f5d5e9... OpenOTP.TokenKey: [PROTECTED BINARY DATA - 20 Bytes] OpenOTP.TokenModel: iPhone10,5 OpenOTP.TokenSerial: 39DEE717-500D-4B31-BF90-A845FC7D81A7 OpenOTP.TokenState: 0 OpenOTP.TokenType: TOTP	Edit
<u>WebADM Settings</u> [delete attribute]	OpenOTP.LoginMode=LDAPMFA,OpenOTP.OTPType=TOKEN OpenOTP.Login Mode: LDAPMFA OpenOTP.OTP Type: TOKEN	Edit

Figure 38. Object Attribute List

Some action buttons appear under the attribute name such as add values or delete the attribute. These actions are determined upon the attribute constraints in the LDAP schema. For example, if an attribute is optional, then you can delete it, and if an attribute can have multiple values, then you can add values or delete some of them.

The attribute value display is dynamically rendered using WebADM attribute type templates (called WebADM attribute handlers). A set of default templates is already defined to display simple data types such as booleans, texts or members as well as complex data types such as certificates, permissions or WebADM-specific data.

After modifying one or several attribute values, you must commit the changes with the *Apply Changes / Delete Selected* button at the bottom of the page. Yet, some special attributes call specific modification pages, which should update the attribute values themselves. This is the case for member lists, permissions, WebADM settings, WebADM data, etc... When an attribute has multiple values, and you want to remove some of them, just select them on the right of the value and click the *Apply Changes / Delete Selected* button at the bottom of the page.

## 6.7 Moving / Copying Objects

WebADM does not provide actions for moving objects. The objects to be moved must be copied and then the original object



should be deleted. To copy from an LDAP server to another, objects can be exported, modified and then re-imported using LDIF.

Copy operations must respect the quotas defined in the OptionSets if you have quotas enabled. Therefore, ensure the copy will not reach your quotas before copying.

It is not recommended to move administrator objects when using certificate-based authentication. Or his certificates must be re-created after moving because the administrator's login DN is part of the certificate data.

Password attributes are invalidated after a copy or import on Novell eDirectory and Microsoft ActiveDirectory. User passwords must also be reset after a copy.

WebADM data inside LDAP objects are encrypted with an AES-256 key and the object DN. The copy action will handle the re-encryption automatically. Yet, an export followed by a re-import at a different place will invalidate the encrypted data.

## 6.8 Exporting / Importing Objects

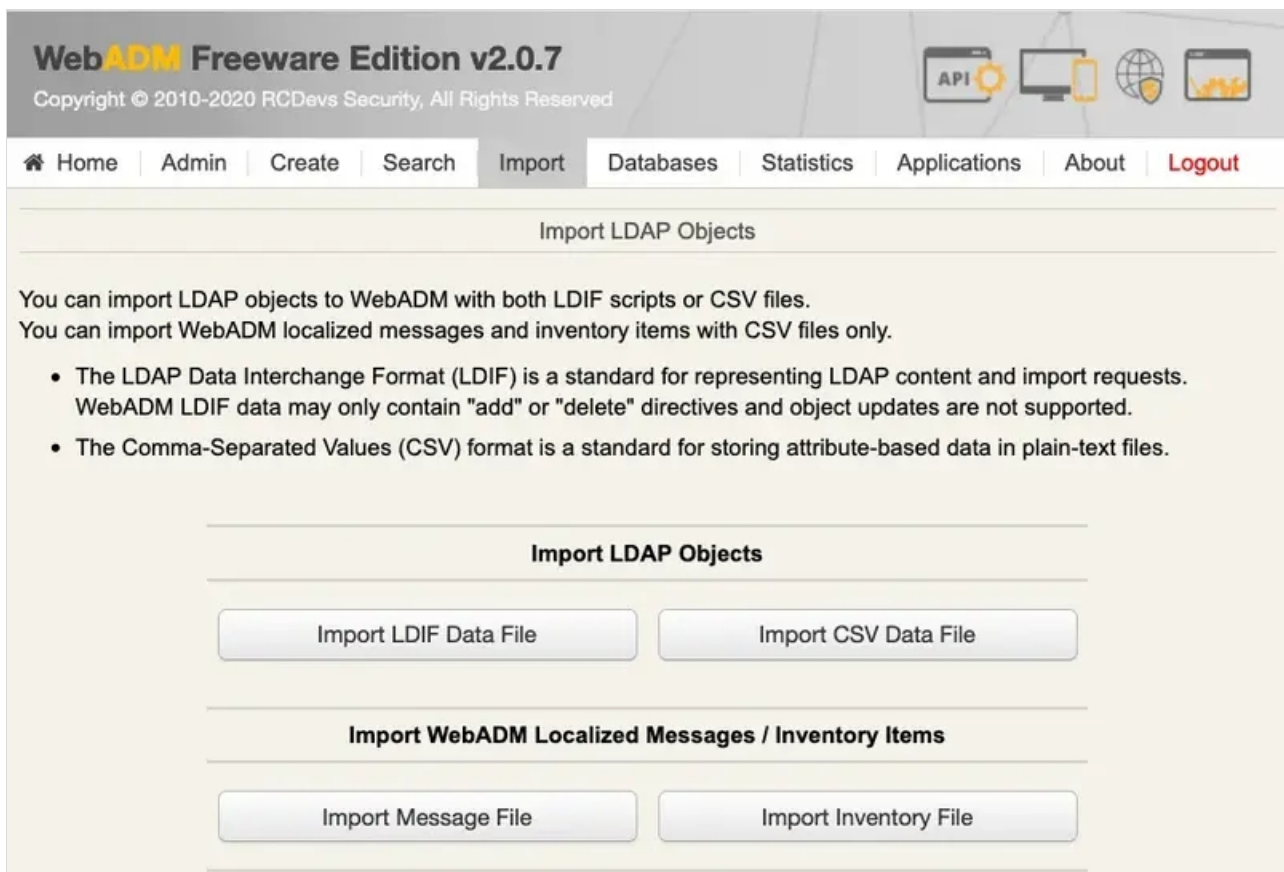


Figure 40a. Import LDAP Objects

### 6.8.1 LDIF Export / Import

WebADM is able to export and import LDAP objects using LDIF files. When editing an object, it is possible to export it or its whole content.

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

### Import LDIF File

Import File:  No file selected.

The LDIF import system only supports *LDAP Add* operations such as in the LDIF files generated by a WebADM export.

Modification or delete LDIF operations will be ignored.

Figure 39. LDIF Import Form

When importing an LDIF file, WebADM operates in two passes:

1. The first pass creates the objects and all their mandatory attributes.
2. The second pass adds the optional attributes.

It is not always possible to create objects in one step because some attribute values may include references to other objects that do not exist at creation time (if they are listed later in the LDIF file). It would not respect the directory integrity and would make it impossible to create some objects. Permissions or group members are some good examples. WebADM allows to export, delete and re-import a subtree with its administrators, permissions and object references by using the two-passes import mechanism.

### Import LDIF File

**Pass 1: Creating objects and mandatory attributes**  
Adding DN: 'cn=Elvis,o=Root'... **Success**

**Pass 2: Adding optional attributes**  
Modifying DN: cn=Elvis,o=Root... **Success**

Figure 40. LDIF Import

With Novell eDirectory and Microsoft ActiveDirectory, the user passwords cannot be restored at import. The password also has to be reset after the import.

For super-administrators, it is possible to export LDAD encrypted attributes (such as webadmData) unencrypted. By default, the LDIF export contains the raw data which is stored in the LDAP directory (with encrypted data). Exporting unencrypted data can be useful for backing up your LDAP users and data.

## Note

The WebADM LDAP encryption depends on the object's DN. If you export users and then re-import them in another location, any encrypted data will be lost. You can use the unencrypted export/import for this purpose.

### 6.8.2 CSV Import

WebADM provides a method for creating a large number of objects of the same type in one single step. The CSV Import feature allows importing a file containing raw object data. The import page asks for the object's type to be created and the creation context. The import file must be structured that way:

- › The first line must contain the attribute names corresponding to the values appearing in the same column in the next lines.
- › The next lines must contain the attribute values for the imported object type. And all the mandatory attributes for the specified object type must be present.
- › The naming attribute must be the first one listed. Fields must be separated by commas.

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

### CSV Import

Import File:  No file selected.

Object Type:

Container:

LDAP object import files must be formatted as below:

1. The first line must contain the attribute names corresponding to the values under the same column in the following lines.
2. The other lines must contain the attribute values for the imported object type.
3. All the mandatory attributes for the specified object type must be present!
4. The LDAPnaming attribute (generally CN) must be listed first.
5. All fields and values must be separated with commas (CSV).

Figure 41. CSV Import Form

### 6.9 Searching for Objects

WebADM search system provides a simple interface to look for objects based on criteria. It works in two modes:

- > The simple search mode allows selecting an attribute, searching criteria and the data to be searched.

The screenshot shows the WebADM Freeware Edition v2.0.7 interface. At the top, there is a navigation menu with links for Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout. The main content area is titled "LDAP Search" and contains a "Simple Search (Advanced Search)" form. The form has the following fields: "Search Base:" with a text input containing "ou=RCDevs,o=Demos" and a "Select" button; "Search for entries whose:" with a dropdown menu set to "Common Name (cn)", a second dropdown set to "contains", and an empty text input; and a "Search" button at the bottom.

Figure 42. Simple Search

- > The advanced search mode provides more detailed searching. You can select the search context and scope, edit the search filter (manually or using the search filter editor) and define what attributes should be searched and displayed.

The screenshot shows the WebADM Freeware Edition v2.0.7 interface. At the top, there is a navigation menu with links for Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout. The main content area is titled "LDAP Search" and contains an "Advanced Search (Simple Search)" form. The form has the following fields: "Search Base:" with a text input containing "ou=RCDevs,o=Demos" and a "Select" button; "Scope:" with a dropdown menu set to "Sub (entire subtree)"; "Filter:" with a text input containing "(objectclass=\*)" and an "Edit" button; and "Return:" with a text input containing "cn,dn,description,fullname,mail,preferredlanguage,langi"; and a "Search" button at the bottom.

Figure 43. Advanced Search

The list of attributes to be searched in the simple mode as well as the attributes to be displayed in the results are configurable in the objects specification file (`conf/objects.xml`).

### 6.9.1 Batch Search Actions

WebADM allows performing batch actions on the resulting entries of a search. The actions that are supported are:

- › Adding *webadmAccount* *objectclass* extension to users.
- › Removing *webadmAccount* *objectclass* extension from users.
- › Adding objects to groups.
- › Removing objects from groups.
- › Setting LDAP attributes.
- › Adding LDAP attribute values.
- › Setting WebADM applications Settings.
- › Removing objects.

The syntaxes and details for each batch action are displayed in the batch actions wizard.

The screenshot shows a web interface titled "LDAP Search" containing an "Advanced Search (Simple Search)" form. The form has the following fields and controls:

- Search Base:** A text input field containing "ou=RCDevs,o=Demos" and a "Select" button to the right.
- Scope:** A dropdown menu currently showing "Sub (entire subtree)".
- Filter:** A text input field containing "(objectclass=\*)" and an "Edit" button to the right.
- Return:** A text input field containing "cn,dn,description,fullname,mail,preferredlanguage,langi".
- Buttons:** Two blue buttons labeled "Search Again" and "Export CSV" are positioned below the "Return" field.
- Batch Action:** A dropdown menu at the bottom of the form is set to "Add WebADM Extension (Activate)", with a "Go" button to its right.

Figure 44. Batch Search Actions

## 7. WebADM OptionSets

Some WebADM restrictions or “subtree options” can be assigned to specific LDAP contexts using WebADM OptionSets. OptionSets are essentially subtree profiles which can be used for example to define a unicity verification context or limiting the LDAP view depth for delegated administrators. Option sets can also be used to create Organization profiles specifying the default LDAP attributes for member objects within the organization. See section WebADM OptionSets for details.

Option sets are used in the WebADM Administrator Portal only and do not interact with the Web Services or WebApps.

When several OptionSets are defined for the same context (even at a different level of the LDAP tree), the options are inherited from the upper tree down to the current context.

All OptionSets must be stored in the same container as specified in the WebADM main configuration file (`conf/webadm.conf`) to be read by WebADM at session startup.

An OptionSet is configured with an LDAP DN which corresponds to the scope of application for the options listed hereafter.

Home Admin Create Search Import Databases Statistics Applications About Logout

Object Settings for `cn=Demos,dc=OptionSets,dc=WebADM`

Disable Option Set       Yes    No (default)

Target Subtree           

The LDAP tree the optionset applies to.

Tree Root Context           

Set a forced LDAP tree view base for any administrators existing inside the target subtree.  
The tree root context will filter SQL audit logs entries based on the user DN in every entry.  
Note: Does not apply for super administrators.

Unicity Check Context           

Context within which unique attributes unicity is verified.

---

LDAP Creation Defaults     

Comma-separated list of default attribute values automatically filled when creating LDAP objects.  
Syntax: Attr1=Value1, Attr2=Value2...

---

**Posix Auto-Increments**

Minimum UNIX UID     

Auto-incremented UID values will start from this value.

Minimum UNIX GID     

Auto-incremented GID values will start from this value.

---

**Mobile Badging**

Office Coordinates           

GPS coordinates used to detect badging from office (ex. 49.502105712890625,5.944442179558995).

Office Networks     

Network(s) with mask to be considered as internal office IP subnets (requires Office Position).  
Web badging from SelfDesk is allowed only from office networks.

Check Badging Expire           

Minimum time for which access remains allowed after a badging in Check mode (in hours).  
If not set, client accesses will remain allowed for one hour anyway.

Check Badging Hours           

Daily hour chunks within which a badging in Check mode remains active.

Badged Users Group

LDAP group to be auto-populated with badged-in users.

Office Users Group

LDAP group to be auto-populated with users badged-in from office.

---

**Remote Work Accounting**

Local Country

The country which should not be considered as remote work in the badging reports.

Remote Quota

Maximum number of remote work days in the selected countries.  
Use a comma-separated list in the form 'FR:32,BE:25' to set per user country quotas.  
Per user country quota requires users to have the country 'c' LDAP attribute set.

---

**User Alert Settings**

User Alerts  Password  Certificate  Badging

Periodically alerts users when passwords or certificates will expire.  
Password near expiration detection works only with ActiveDirectory.  
Badging sends a warning to users who forgot to badge-out yesterday.

Alert Period

Start sending alerts 1 to 30 days before expiration.

Alert Repeat

Re-send alert messages every 1 to 5 days.

Figure 45. OptionSet Configuration

- > **Tree Root Context**: Set the tree view base for administrators. This option is mainly used to limit the administrators' LDAP access scope in WebADM when using OpenLDAP and Microsoft ActiveDirectory. With Novell eDirectory, the tree access limitation is provided by the LDAP permissions (ACL) which can be set directly on LDAP container nodes. The tree root option prevents administrators from accessing any object out of the specified tree base and reduces the tree view accordingly.

The root context applies to the SQL logs viewer too where the log events corresponding to objects outside the root context are filtered and not displayed.

## Note

This option concerns administrators only and is computed at login time. Any administrator located inside the configured target subtree will also have its LDAP tree view limited to the configured tree root context.

- > **Unicity Context**: Defines the LDAP tree base to be used by WebADM for checking unique users' attributes. Unique attributes are defined in the objects specification file (`conf/objects.xml`).

The unicity context is used for other purposes in WebADM such as the tree base for determining free UID numbers for POSIX accounts.

- > **LDAP Defaults**: List of default attribute values which will be autofilled when creating or extending objects inside the target subtree.

- > **Minimum UNIX UID** : Auto-incremented UID values will start from this value.
- > **Minimum UNIX GID** : Auto-incremented GID values will start from this value.
- > **Office Coordinates** : Defines the GPS coordinates for the mobile badging option to badge from these office coordinates.
- > **Office Networks** : This parameter requires office position. It defines internal office IP subnets from which users are allowed to proceed to badge from SelfDesk.
- > **Check Badging Expire** : Time during which the badging is effective in hours. By default, it is valid for one hour.
- > **Check Badging Hours** : Time slots during which the badging remain possible. When not in these time slots, users can't badge at all.
- > **Badged Users Group** : Groups selected to be populated by users who have badged-in.
- > **Office Users Group** : Same groups as parameter before but for users badged-in from the office. Requires Office coordinates.
- > **Local Country** : The country which will not be considered as remote for the users who badge there.
- > **Remote Quota** : The maximum number of days of remote work possible per country. Defined by the country code, followed by a column and the number of days : 'FR:32,BE:25,LU:45'. The users need to have the country defined in their LDAP attributes or the parameter won't be effective.
- > **User Alerts** : 3 options that are Password, Certificate and Badging. The first two concerns an expiration that is soon, so it alerts the user. Regarding the password, it works only with Active Directory. The badging alert is for a user who forgot to badge-out the day before.
- > **Alert Period** : The number of days the alert is sent before the expiration. 10 days by default.
- > **Alert Repeat** : The number of days the alert is repeated after the first notification that the expiration is soon. 3 days by default.

## 8. WebADM AdminRoles

WebADM includes the concept of delegated administration. It also makes the distinction between Super Administrators and Other Administrators. Super Administrator is an LDAP administrator (ex. AD Domain Admin users) which are configured in the *super\_admins* list in conf `/webadm.conf`. The Super Administrators have unlimited access to any feature of WebADM. On the contrary, Other Administrators are the delegated administrators for which you can define precisely what features and administration operations are allowed through WebADM AdminRole objects. Another Administrator is also any LDAP user which is a member of one or several WebADM AdminRole(s).

LDAP access rights for both Super Administrators and Other Administrators MUST be set at the LDAP server level with dedicated LDAP ACLs. This is important to notice that WebADM enforces access control over its own management interfaces, but it cannot enforce any security control at the LDAP API level! This means that restricting user operations and features via AdminRole configurations does not prevent an administrator from performing the same operations from another LDAP client software.

All AdminRoles must be stored in the same container as specified in the WebADM main configuration file (`conf/webadm.conf`) to be read by WebADM at session startup.



An AdminRole can be applied to a single administrator account or a group of administrators (but nested groups are not supported).

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

Object Settings for **cn=Other\_Admins,dc=AdminRoles,dc=WebADM**

Disable Admin Role     Yes     No (default)

Assigned User or Group       

Selected user or group of LDAP administrators which is assigned this role template.  
Note: All the restrictions below do not apply for super administrators.

Figure 46. AdminRole Assigned Group

## 8.1 Basic Permissions

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

**Basic Permissions**

Allowed Interfaces     Admin (Default)     Manager  
Allow access to the Admin Portal, the Manager RPC interface or both.

Created Objects          
Comma-separated list of LDAP objects which can be created under the 'Create' menu.  
The same object types can be copied, imported and deleted respecting management rights.

Allowed Configurations          
Comma-separated list of configuration objects which can be managed under the 'Admin' menu.  
Tree access to the WebADM configuration container is required for managing configurations.

Allowed Databases          
Comma-separated list of SQL tables accessible under the Database menu in read-only mode.

Managed Databases          
Comma-separated list of SQL tables accessible with edit, import and delete permissions.

Allowed Log Files          
Comma-separated list of log files accessible under the Database menu.

**Detailed Permissions**

Figure 47. AdminRole Basic Permissions

- > **Allowed Interfaces** : Controls which administration interface is available for the selected administrator(s). Admin enables access to WebADM Admin Portal. Manager provides access to the JSON-RPC management interface. By default, access to the Manager interface is denied.
- > **Created Objects** : Contains a list of object classes defining which LDAP object types can be created, imported and deleted. Any LDAP object containing at least one of these allowed object classes are authorized for creation, import and deletion.
- > **Allowed Configurations** : Defined the list of configuration objects which can be managed under the 'Admin' menu. Note that graphical access (i.e. browsing capability) to the WebADM configuration containers is required for managing WebADM configurations. This setting enables restrictions to the configuration objects when accessed from WebADM but does not prevent an administrator from editing the corresponding LDAP objects from another LDAP interface.
- > **Allowed Databases** : Defines which SQL database tables (logs, localized message and inventory) are accessible. The selected database tables are accessible in read-only by default.

#### Note

This option does not apply for super administrators.

- > **Managed Databases** : Defines which SQL database tables (log, localized message and inventory) are accessible in write or edition mode. For logs, write access provides deletion of selected entries and purge of old events. For Message and Inventory, write access provides import and management of entries.
- > **Allowed Log Files** : Defines which WebADM log files are accessible under the Database menu.
- > **Allowed Applications** : Defines which applications (WebApps and Web Services) are configurable by the administrators.

#### Note

This option does not apply for super administrators.

## 8.2 Management Rights

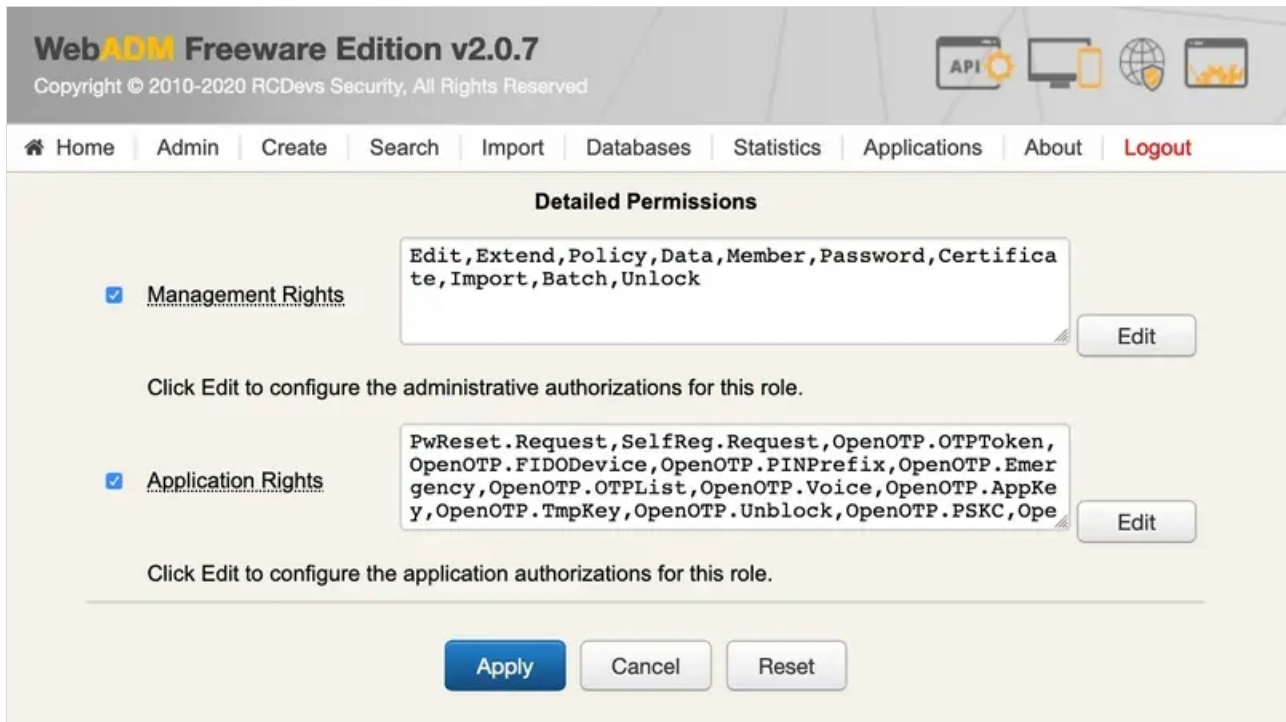







Figure 48. AdminRole Detailed Permissions

By clicking the Edit button on the right side of the management rights, you can configure what LDAP object management features and WebADM operations are allowed. By default, none of the listed rights is enabled.

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

API     


Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

**LDAP Management Features**

---

**Basic Objects Operations**


Add, remove, modify attributes. Delete and rename objects.

Note: Role features are used to enable WebADM management options in the interface and should not be considered a security limitations. LDAP ACL rights must also be setup accordingly! 

---


**Add Object Extensions**

Add and remove auxilliary object classes with their attributes.

This role also provides user activation which consists in adding the WebADM extension on the users and group activation which is required if you need group-based policies. 


---

**Edit User & Group Policies**

Configure application settings on activated users and groups. 


---

**Edit User Application Data**

Manually edit application data on activated users and groups. 


---

**Manage Group Membership**

Add users to group objects and set group membership on the user objects. 


---

**Change User Passwords**

Initialize and reset user passwords. 

---

**Create User Certificates**

Create and renew user certificates for use in WebApps.  
Create and renew admin certificates for login in PKI mode. 

---

**Additional Management Features**

Figure 49. AdminRole LDAP Management Rights

- > **Basic Object Edition**: Modify attributes, rename and remove LDAP objects.
- > **Add Object Extensions**: Add and remove auxiliary object classes with their attributes. This role also provides user activation which consists of adding the WebADM extension on the users and group activation which is required if you need group-based policies.
- > **Edit User & Group Policies**: Configure application settings on activated users and groups.
- > **Edit User Application Data**: Manually edit application data on activated users and groups.
- > **Manage Group Membership**: Add users to group objects and set group membership on the user objects.

- > **Change User Passwords** : Initialise and reset user passwords.
- > **Create User Certificates** : Create/renew user certificates to be used in WebApps and create/renew admin certificates to be used for login when WebADM is configured in PKI mode.

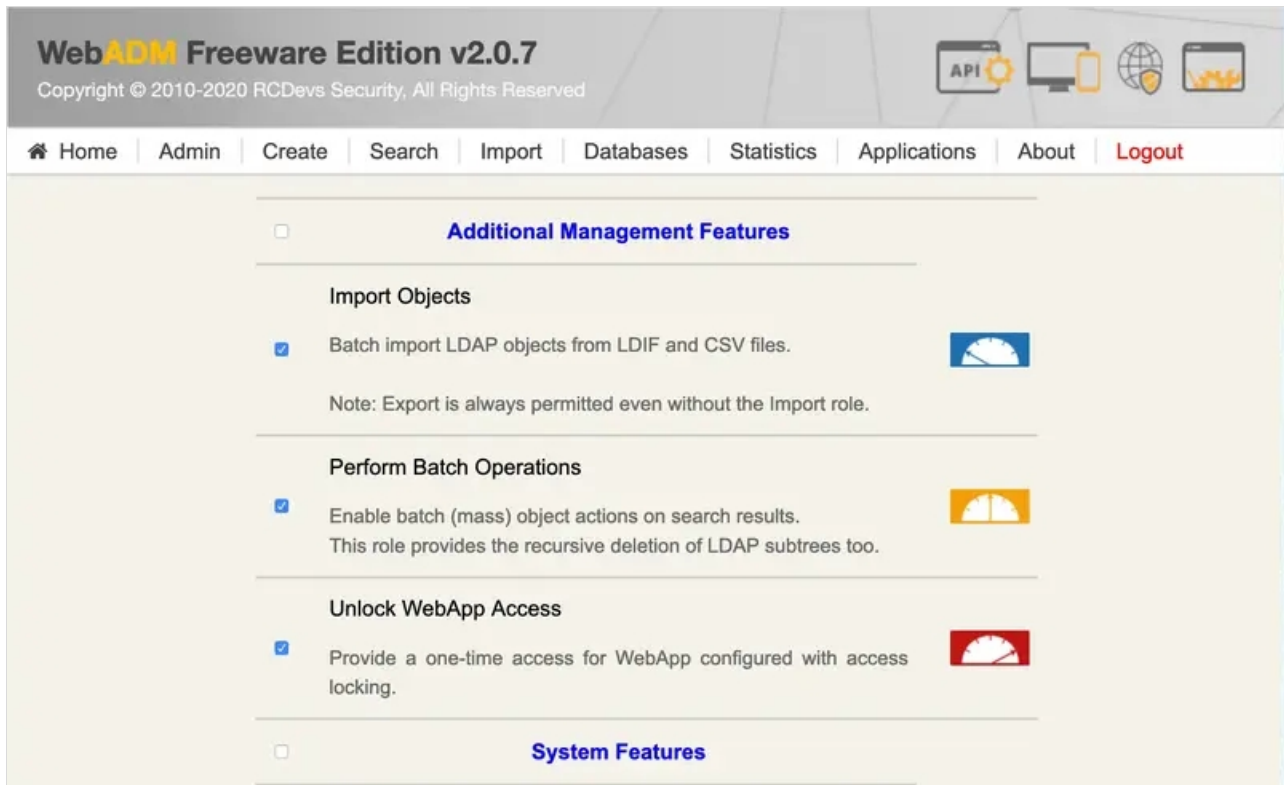


Figure 50. AdminRole Additional Management Rights

- > **Import Objects** : Batch import LDAP objects from LDIF and CSV files.

#### Note

Export is always permitted even without the Import role.

- > **Perform Batch Operations** : Enable batch (mass) object actions on search results. This role provides the recursive deletion of LDAP subtrees too.
- > **Unlock WebApp Access** : Provide one-time access for WebApp configured with access locking.

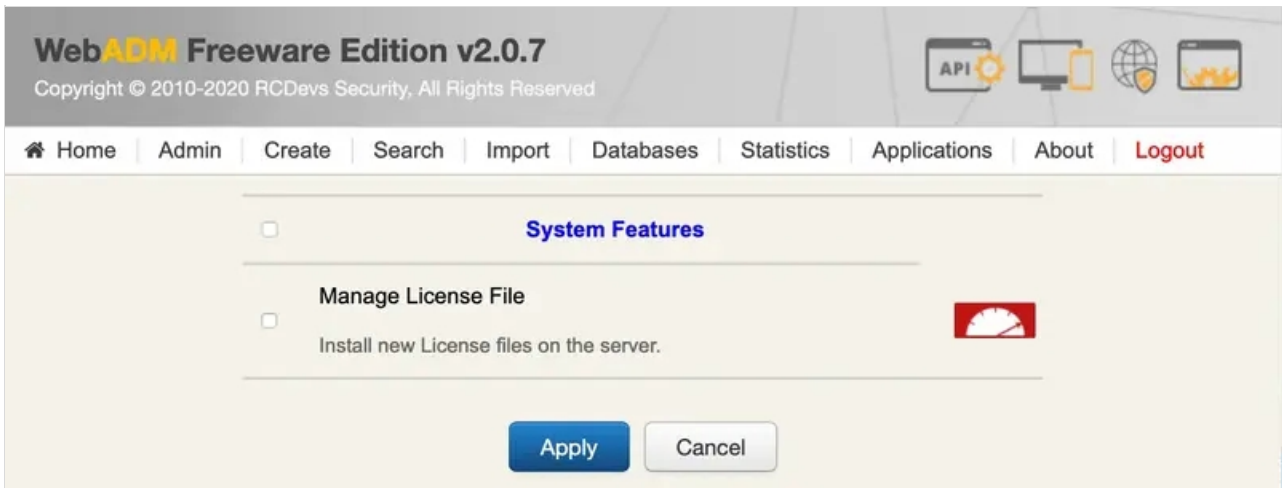


Figure 51. AdminRole System Rights

> **Manage License File**: Install new licenses on the server from the WebADM Admin Portal.

### ⚠ Important

The above role features should be used to restrict WebADM management options for delegated administrators but should not be considered for hard security limitations. LDAP ACL rights must also be setup accordingly!

## 8.3 Application Rights

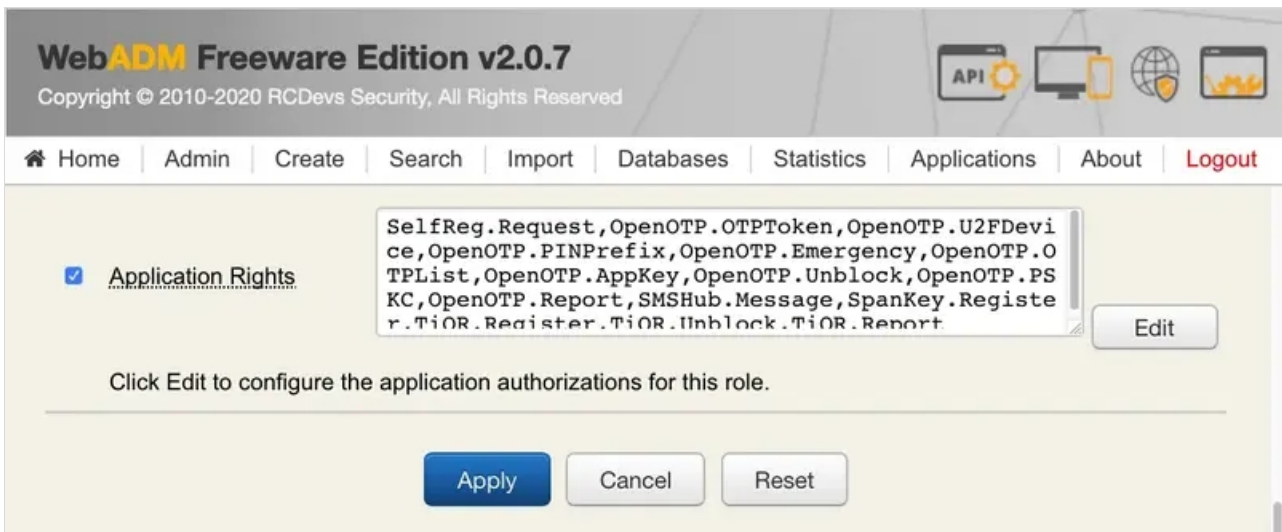
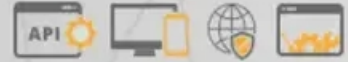


Figure 52. AdminRole Application Rights

The AdminRole object provides full control over which application operations and features are allowed for the delegated administrators. Any registered application like OpenOTP or the Self-Services provides a list of role-based authorizations which can also be assigned to an AdminRole.



Application Rights

Secure Password Reset

Send Email & SMS Request

Send password requests to a single user or a group of users.

User Self-Registration

Send Email & SMS Request

Send enrollment requests to a single user or a group of users.

MFA Authentication Server

Manage OTP Tokens

Register and un-register Software / Hardware Tokens and Printed OTP Lists.

Disable, enable and resynchronize Tokens, set OCRA PIN codes.

Manage FIDO Devices

Register and un-register FIDO Devices.

Manage OTP PIN Prefix

Manage Emergency OTP

Manage Printed OTP List

Manage Voice Biometrics

Manage Application Passwords

Manage Temporary Passwords

This feature is accessible from the Manager interface only.

Unblock Blocked Accounts

Import & Export PSKC Files

Register Tokens from PSKC files and export to encrypted PSKC.

Enable Reporting Methods (Manager Only)

This permissions is for the Manager Interface only.

**SMS Hub Server**

**Send SMS Messages**  
Send text messages to a single user or a group of users.

**SSH Public Key Server**

**Manage SSH Keys**  
Register and un-register SSH keys, update key expiration dates.

**Apply** **Cancel**

Figure 53. AdminRole Sample Application Rights

The role-based permissions are per-application and their documentation is not in the scope of this document. Check the online documentation for your registered applications for details.

## 9. WebADM LDAP MountPoints

MountPoints are containers in the LDAP tree that include objects and child containers (i.e. the entire tree structure) from another LDAP server. The objects are not physically present in the main tree structure. Instead, WebADM connects at runtime to an external LDAP and renders its contents as if the data were stored in the mounting context.

MountPoints are LDAP objects that hold LDAP connection parameters. WebADM connects to a remote LDAP using these parameters. Once connected, WebADM retrieves the remote LDAP tree structure and renders it on the main tree. The rendering location, or namely the mounting point, is specified inside the MountPoint object.

All MountPoints must be stored in the same container (specified in the WebADM configuration file) to be read by WebADM at start-up.

WebADM provides a virtual DN notation for accessing objects in mounted LDAP trees. This notation concatenates the MountPoint DN (of the main LDAP) with the real DN in the mounted LDAP and works with all the pages having DN inputs. You can check the virtual DN when editing an object in a mounted LDAP.

The MountPoint supports the following settings:

- > **Mount DN**: The local LDAP tree node where the mounted LDAP tree should be mounted.
- > **Host Name**: The hostname or IP address of the mounted LDAP server.
- > **Port Number**: The LDAP port number of the mounted LDAP server.
- > **Connection Type**: The connection type used to connect the mounted LDAP server. Allowed connection types are None (no encryption), SSL and TLS.
- > **Tree Base**: The tree base on the mounted LDAP server.
- > **Login DN**: The DN used to bind the mounted LDAP server. This account must have write permissions on the mounted LDAP server. An empty login DN means anonymous LDAP bind.



- > **Login Password** : The password corresponding to the login DN.
- > **Client Certificate File** : The client certificate if the mounted LDAP server requires certificate-based client authentication.
- > **Client Certificate Key File** : The certificate key corresponding to the client certificate.

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home Admin Create Search Import Databases Statistics Applications About Logout

Object Settings for **cn=Novell eDirectory,dc=MountPoints,dc=WebADM**

**Disable Mount Point**       Yes  No (default)

**Mount DN**              
The LDAP tree node where to mount the remote LDAP.

**Host Name(s)**        
LDAP server name(s) or IP address(es).  
You can set a comma-separated list of servers. The next servers are used for failover.

**Port Number**        
LDAP server port.

**Encryption Type**     

**Tree Base**        
Mounted LDAP tree base or base DN (mandatory with most LDAP servers).

**Login DN**        
Mounted LDAP bind DN. WebADM will bind anonymously if not set.

**Login Password**     

**Trusted CA Certificate**           

**Client Certificate File**           

**Client Certificate Key File**           

Figure 54. LDAP MountPoint Settings

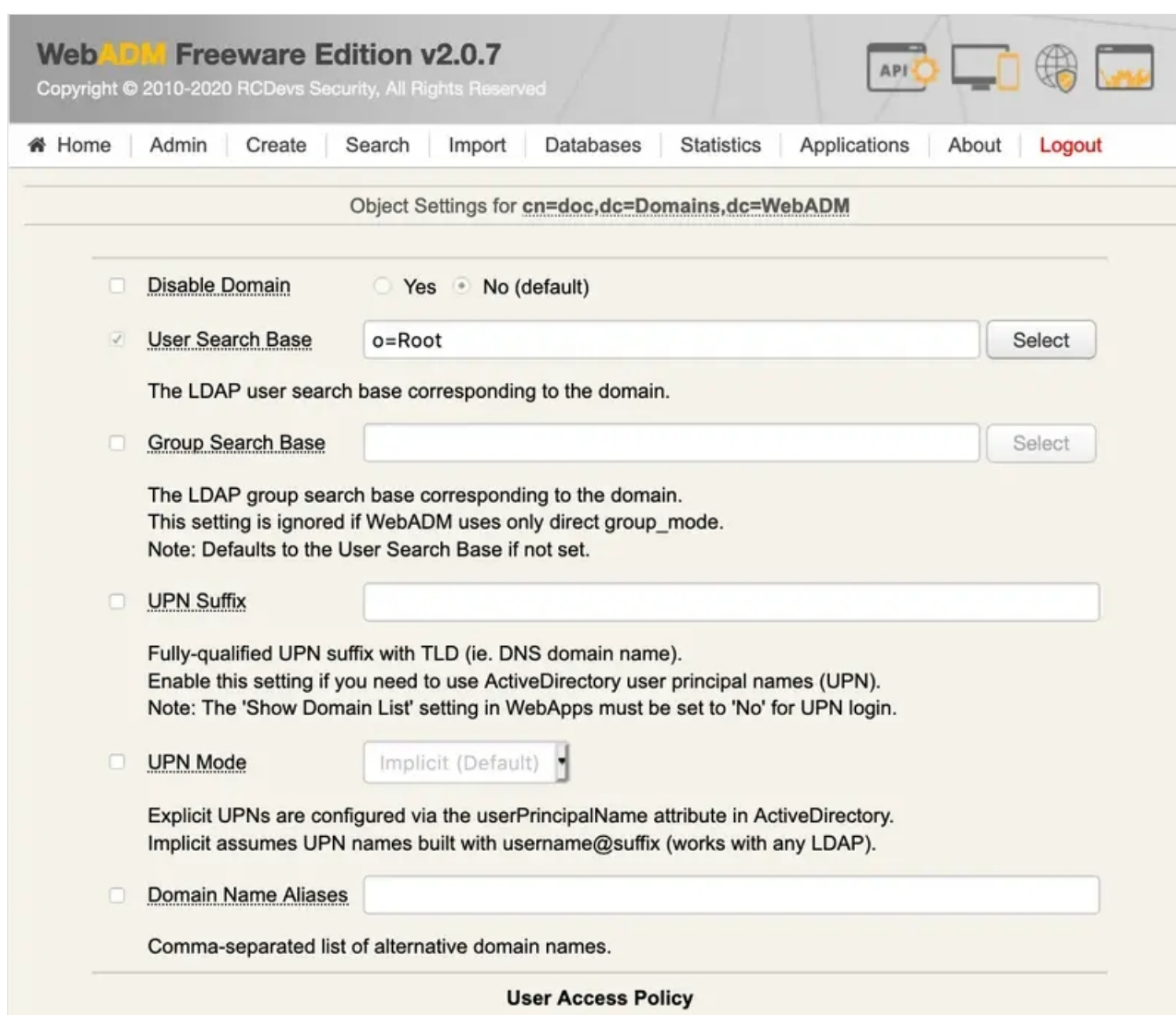
## 10. WebADM LDAP Domains

WebADM Domains are used by the registered WebADM applications to identify a user with a username, a password and a domain name. The domain objects establish the relationship between a domain name and an LDAP tree base. Also, when an application wants to obtain an LDAP user DN corresponding to the provided login information, it will use the domain tree base to build the LDAP search.

All Domains must be stored in the same container (specified in the WebADM configuration file) to be read by WebADM at start-up.

A WebADM Domain object supports the following settings:

- > User Search Base: The tree base corresponding to the domain and to be used in the user LDAP searches.
- > Group Search Base: The tree base to be used in the LDAP group searches. If not specified, it defaults to the Tree DN. This setting will be ignored if WebADM is configured to use direct groups only.
- > Domain Name Aliases: A comma-separated list of aliases for the Domain name. Setting multiple names for a single Domain can be useful in the following scenarios:
  - > You want to enable both ActiveDirectory, NetBIOS and DNS domains naming for your integrations (ex. MYCOMPANY and mycompany.com).
  - > You use ActiveDirectory User Principal Names (UPNs). In this case, you can create a Domain alias corresponding to the users' UPN suffix.



**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security, All Rights Reserved

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

Object Settings for cn=doc,dc=Domains,dc=WebADM

Disable Domain       Yes    No (default)

User Search Base              
The LDAP user search base corresponding to the domain.

Group Search Base              
The LDAP group search base corresponding to the domain.  
This setting is ignored if WebADM uses only direct group\_mode.  
Note: Defaults to the User Search Base if not set.

UPN Suffix        
Fully-qualified UPN suffix with TLD (ie. DNS domain name).  
Enable this setting if you need to use ActiveDirectory user principal names (UPN).  
Note: The 'Show Domain List' setting in WebApps must be set to 'No' for UPN login.

UPN Mode        
Explicit UPNs are configured via the userPrincipalName attribute in ActiveDirectory.  
Implicit assumes UPN names built with username@suffix (works with any LDAP).

Domain Name Aliases        
Comma-separated list of alternative domain names.

**User Access Policy**

Figure 55. LDAP Domain Settings

The Domain User Search Base can be set to a container inside a mounted LDAP or the LDAP mount point DN itself (see MountPoints). This is a very convenient way to assign a domain to the users of another LDAP server.

## Important

The WebADM Domains are not the same thing as the LDAP Domain containers (example: dc=myobject). LDAP Domain containers (DC objects) are generic containers like Organizations or Organizational Units. Whereas WebADM Domains are objects of type webadm\_config\_object such as the OptionSets, MountPoints or Clients and contain some settings.

A WebADM Domain object supports the following user access policy settings:

- > **Allowed Groups**: Mandatory LDAP group(s) the domain users must belong to (in order to be considered as part of the domain). If set, users must be a member of at least one of the listed groups.
- > **Excluded Groups**: Exclusion LDAP group(s) the domain users must not belong to. If set, users must not be a member of the listed groups.
- > **Allowed Addresses**: Required network address(es) with netmask the domain must be accessed from. If set, users must be located in at least one of the listed networks (ex. 192.168.1.0/24).
- > **Excluded Addresses**: Excluded network address(es) with netmask the domain must not be accessed from. If set, users must not be located in any of the listed networks.
- > **Allowed Locations**: Required country code(s) the domain must be accessed from. If set, users must be located in at least one of the listed countries.
- > **Excluded Locations**: Excluded country code(s) the domain must not be accessed from. If set, users must not be located in any of the listed countries.
- > **Allowed Hours**: If set, the domain can be used only during the specified week hours.
- > **Excluded Hours**: If set, the domain cannot be used during the specified week hours.

**WebADM Enterprise Edition v2.0.7**  
 Copyright © 2010-2020 RCDDevs Security, All Rights Reserved

API | [Monitor] | [Mobile] | [Globe] | [Bar Chart]

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

---

**User Access Policy**

Allowed Groups

Required LDAP group(s) the domain users must belong to (one per line).  
 If set, users must be a member of at least one of the listed groups.

Excluded Groups

Excluded LDAP group(s) the domain users must not belong to (one per line).  
 If set, users must not be a member of any of the listed groups.

Allowed Addresses

Comma-separated list of IP addresses with netmasks the domain must be used from.  
 If set, the application must be accessed from the listed networks (ex: 192.168.1.0/24).

Allowed Locations

Comma-separated list of country codes the domain must be used from.  
 If set, users must be located in at least one of the listed countries.

Allowed Hours

If set, the domain can be used only during the specified week hours.

Excluded Days

If set, the domain cannot be used during the specified days.

---

**Application Access Policy**

Figure 56. LDAP Domain User Access Policy

Locations and Hours should set graphically using the Edit buttons on the right side.

A WebADM Domain object supports the following user access policy settings:

- > **Allowed WebApps / Web Services**: List of applications with which the domain is used. By default, a domain works with all registered applications (Web Applications and Web Services).

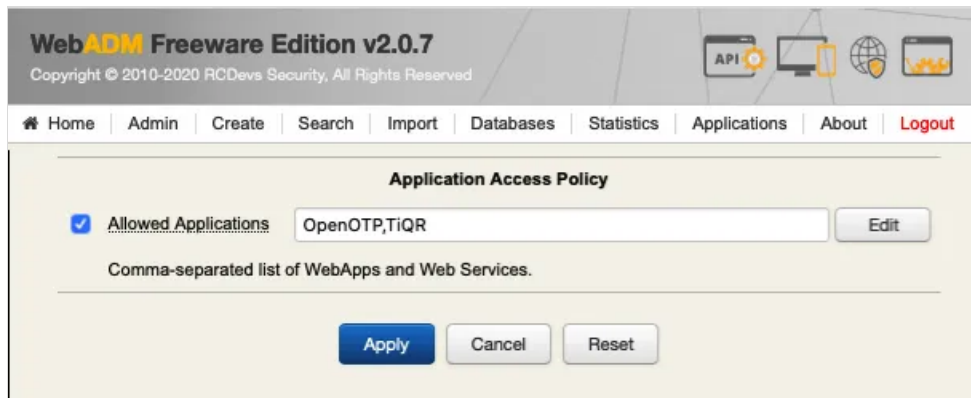


Figure 57. LDAP Domain Application Access Policy

## 11. WebADM Client Policies

A Client Policy object can be defined if you need to define per-client applications' access control policies or if you want to force some Web Service settings for the client applications. A client application means a remote system which uses the SOAP or RADIUS APIs. You can also define per-client application profiles or policies in WebADM by using Client objects.

By defining a Client object, you can, for example, restrict access to the Client application for some LDAP authorized groups or prevent some groups to use the application. You can even restrict the application access to some WebADM Domains.

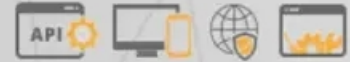
Another feature of the Client is that you can define some Web Application settings which will always be enforced for the Client application whatever setting is set in the users or its groups. For example, you want one VPN to authenticate users through RCDevs OpenOTP with LDAP+OTP passwords and Token whatever policy is defined for the user, and you want your internal systems to authenticate users with LDAP only.

To create a Client profile, you must know your client application IDs. The WebADM Client object must have the same name as the client ID. The ID is typically the Client name that appears in the WebADM Log Viewer for Web Services. The Client ID is generally provided in the client requests in the client SOAP attribute. With RADIUS, it is the NAS-Identifier. If this information is not provided by the client, WebADM will use the client IP Address as the client name.

A WebADM Client supports the following user access policy settings:

- > **Disable Client**: Enables or disables the Client profile.
- > **Default Domain**: The Web Services SOAP APIs under WebADM support multiple Domains. When the Client does not provide the user domain name in the SOAP requests, WebADM will look at the targeted Web Service configuration for a default Domain. But if the Client object corresponding to that request has a default Domain set, it will be used in priority.
- > **Friendly Name**: The friendly Client name is a short description to be used in the application's user messages which contain a %CLIENT% variable.
- > **Client Name Aliases**: You can define a comma-separated list of aliases for your client name. Defining Client aliases is useful with OpenOTP RadiusBridge when your RADIUS client (NAS) does not support passing the Client ID via the NAS-Identifier attribute (Ex. Cisco ASA). In this case, you need to use the NAS IP address as Client name in order to define a Client Policy. With the alias, you can define the Client with a name of your choice and simply set the NAS IP Address as an alias.
- > **Allowed Domains**: You can restrict the Domains that are usable for the client application.

- > **Allowed Groups** : Another possibility is to restrict the client application access based on the user groups. Only the users part of at least one of the listed groups will be authorized.
- > **Excluded Groups** : You can define a set of groups which cannot use the client application. If a user is a part of at least one of the excluded groups, then he cannot use the client application.
- > **Allowed Addresses** : Required network address(es) with netmask the client application must be accessed from. If set, users must be located in at least one of the listed networks (ex. 192.168.1.0/24).
- > **Excluded Addresses** : Excluded network address(es) with netmask the client application must not be accessed from. If set, users must not be located in any of the listed networks.
- > **Allowed Locations** : Required country code(s) the client application must be accessed from. If set, users must be located in at least one of the listed countries.
- > **Excluded Locations** : Excluded country code(s) the client application must not be accessed from. If set, users must not be located in any of the listed countries.
- > **Allowed Hours** : If set, the client application can be used only during the specified week hours.
- > **Excluded Hours** : If set, the client application cannot be used during the specified week hours.



## Object Settings for cn=Demos,dc=Clients,dc=WebADM

Disable Client       Yes  No (default)

When disabled, client requests using this client policy will be refused.

Default Domain     

This domain is automatically selected when no domain is provided.

Friendly Name     

Friendly client name or short description to be used for %CLIENT% in user messages.

Client Name Aliases     

Comma-separated list of alternative client IDs.

UID Attributes     

Restricted list of LDAP login attributes replacing the attributes configured via uid\_attrs in webadm.conf.

### User Access Policy

Allowed Domains     

List of authorized domains. If not set, any domain is allowed.

Allowed Groups     

Required LDAP group(s) the users must belong to (one per line).  
If set, users must be a member of at least one of the listed groups.

Excluded Groups     

Exclusion LDAP group(s) the users must not belong to (one per line).  
If set, users must not be a member of any of the listed groups.

Allowed Addresses     

Comma-separated list of IP addresses with netmasks the client must be used from.  
If set, the application must be accessed from the listed networks (ex: 192.168.1.0/24).

**Allowed Locations**

Comma-separated list of country code(s) the client must be used from.  
If set, users must be located in at least one of the listed countries.

**Allowed Hours**

If set, the client can be used only during the specified week hours.

**Excluded Days**

If set, the client cannot be used during the specified days.

**Required Attributes**

Required LDAP attribute values the users object must contain in value-pair format.  
Example: ou=unit1,ou=unit2,mobile=+33\*

---

**Forced Application Policies**

Figure 59. Client User Access Policy Settings

A WebADM Client can be configured to enforce specific Web Service settings for the client application.

- > **Application Settings (Default)** : You can configure some Web Service settings which will override any default, user or group setting. Request settings (if present) will still override the forced application settings. Example:  
OpenOTP.LoginMode=OTP,OpenOTP.OTPTType=TOKEN
- > **Group List** : You can set a list of LDAP groups for which you need dedicated application settings (overriding the Default Application Settings defined above).
- > **Application Settings (Group)** : If the users belong to the groups defined above, then these Group Application Settings are prioritized.
- > **Internal Networks** : You can set a list of IP addresses with netmasks corresponding to your internal or trusted network(s).
- > **Application Settings (Internal)** : If the client is used from the internal network(s), then these Internal Application Settings are prioritized.



### Forced Application Policies

Application Settings (Default)

OpenOTP.TrustedContext=No  
 OpenOTP.GeoFence=No  
 OpenOTP.AppKeys=Yes  
 SpanKey.X11Forwarding=No  
 SpanKey.PortForwarding=No  
 SpanKey.AgentForwarding=No

List of application settings which override any default, user or group level setting.  
 The format is the same as for the web services' request settings (see API documentation).  
 The request settings (if present) will still override the application settings.  
 Enter one setting per line in the form OpenOTP.LoginMode=OTP.

---

### Per-Group Extra Policies

Group List

List of LDAP groups with dedicated settings (override any defined Application Setting).

Application Settings (Group)

If the users belong to one of the above group(s), these additional settings are enforced.

---

### Per-Network Extra Policies

Internal Networks

Comma-separated list of IP addresses with netmasks corresponding to your internal network(s).

Application Settings (Internal)

If the client is used from the above internal network(s) these additional settings are enforced.  
 If both Group and Internal settings are enforced, Network settings apply last (higher weight).

*Figure 59. Figure 60. Client Application Policy Settings*

In order to know the syntax for the application settings, you can go to the Application menu, then configure one application. Put the mouse over one of the setting names and the real setting will appear. For example, OpenOTP Login Mode will display LoginMode (public list). The setting name must be prefixed by the Web Service Name and a separating dot. And only the public settings can be set in a Client object.

**Note**

SOAP requests are also able to transport user settings in order to implement per-application settings. Request settings have the highest priority and will override any forced settings.

## 12. Log Viewer

The WebADM SQL logs are accessible from the Databases menu. By default, WebADM provides the following logs:

- > **The Admin Logs** : Contains all the actions performed by administrators in the WebADM Administrator Portal. It also contains all operations performed via the WebADM Manager interface.
- > **The WebApp Logs** : Contains all the actions reported by the WebApps registered in WebADM.
- > **The Web Services Logs** : Contains all the actions reported by the Web Services registered in WebADM.
- > **The Alert Log** : Contains all the error events reported by the Web Applications and Web Services.

The screenshot displays the WebADM Freeware Edition v2.0.7 interface. The top navigation bar includes Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout. The main content area is titled 'Database Viewer for WebApp Logs (1000 results out of 2230 log items)'. It features a search and filter section with 'Event Time' set to 'Equals' and a date range from 'This Minute' to 'This Month'. Below this are 'Display Options' (Retrieve max: 1000, Page results: 35), 'Log Actions' (Delete selected items, Export as CSV/XML, Statistics as CSV/XML, Draw source map), 'Statistic Options' (Show first: ALL, Group by: None), and 'Database Pruning' (Delete log entries older than: 6 Month, Clean). The main table shows log events with columns for Event Time, Application, User DN, User IP, Session ID, and Details. A map overlay is visible over the table, showing a red pin in Paris, France.

Event Time	Application	User DN	User IP	Session ID	Details
2018-12-19 15:12:37	DemoReg	cn=baqyhu,ou=users,o=demos	185.19.222.110	NQOYY0QY	New demo user created (baqyhu)
2018-12-19 14:25:22	SelfDesk	cn=FGTH,ou=Users,o=Demos	84.14.179.226	V47HMREE	Modified user infos (mobile)
2018-12-19 14:23:45	SelfDesk	cn=FGTH,ou=Users,o=Demos			PFallback to SMS
2018-12-19 14:23:39	SelfDesk	cn=FGTH,ou=Users,o=Demos			Type to SMS
2018-12-19 14:23:26	SelfDesk	cn=FGTH,ou=Users,o=Demos			PFallback to SMS
2018-12-19 14:23:06	DemoReg	cn=FGTH,ou=users,o=demos			PFallback to SMS
2018-12-18 21:56:30	SelfDesk	cn=hwilsonnz,ou=Users,o=Demos			PFallback to SMS
2018-12-18 21:56:19	DemoReg	cn=hwilsonnz,ou=users,o=demos			PFallback to SMS
2018-12-18 18:42:42	SelfDesk	cn=jgmanville,ou=Users,o=Demos			PFallback to SMS
2018-12-18 18:42:39	SelfDesk	cn=jgmanville,ou=Users,o=Demos			PFallback to SMS
2018-12-18 18:41:45	SelfDesk	cn=jgmanville,ou=Users,o=Demos			PFallback to SMS
2018-12-18 18:41:41	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	TS41S7ET	Disabled OpenOTP Push Login
2018-12-18 18:37:41	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	TS41S7ET	Login to moux
2018-12-18 18:35:09	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	RVAJ9ZJ7	Login to moux
2018-12-18 18:35:08	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	NI4HVERS	Logged out
2018-12-18 18:34:21	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	NI4HVERS	Resetted OpenOTP PushLogin to Enabled
2018-12-18 18:31:11	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	NI4HVERS	Registered TOTP Push Token
2018-12-18 18:30:54	SelfDesk	cn=jgmanville,ou=Users,o=Demos	96.234.151.187	NI4HVERS	Unregistered TOTP Token

Figure 61. Log Events

Important: If you define an OptionSet with a Tree Base restriction, the same restriction will apply to the log entries to be displayed in the log viewer. So the administrators will only see the user action logs corresponding to user DNs under their own scope of visibility.

By default, all log entries found in the WebADM logs database are shown. If the number of logs is large, you can create log filters

to narrow down the number of logs shown on the screen.

## 12.1 Creating Log Filters

You can create filters with different criteria. There are a number of filters types that can be combined with operators and the searched value to produce accurate filtering results. You can also filter the logs by time, administrator names, session IDs, application names... You can click the links in the log items to generate automatic filters too.



Figure 62. Log Filters

## 12.2 Log Display Options

By default, all log entries found in the WebADM logs database are shown. If the number of logs are large, you can narrow down the number of logs shown on the screen. You can control the number of visible logs in the area using:

- › The Retrieve last value controls the number of last log entries retrieved from the logs database.
- › The Per page results value controls the number of log entries shown per page. The log results are paginated, and you can switch page with the links at the bottom right of the page.

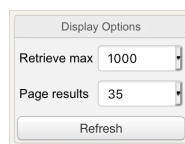


Figure 63. Log Display Options

## 12.3 Log Result Actions

A number of log result actions is available to you.

- › The Delete Selected deletes the logs selected by checking the checkbox next to the log entry in the log result list.
- › The Export (CSV) link exports the selected logs to comma separated value text (CSV) file. You can save this file and view it in the application of your choice.
- › The Statistics (CSV) link creates comma separated value text (CSV) statistics of the selected log column. You can select the column by checking the checkbox in the appropriate column title in the log result list. You can define the number of entries in the statistics with the Display First value. You can get statistics grouped by time steps with the Group By value.

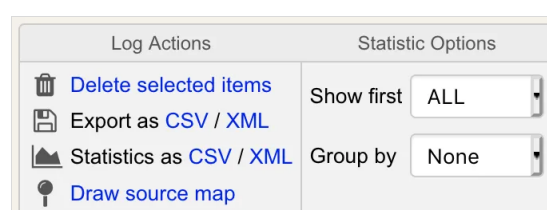


Figure 64. Log Display Options

## 12.4 Pruning the Log Database

In time, the log database grows, and you have to prune it. Enter the pruning time values in the data entry fields press the Clean button to delete logs older than specified from the log database.

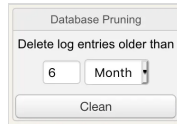


Figure 65. Log Database Pruning

## 12.5 Source Map Viewer

With the log viewer, you can graphically draw your current selection of user accesses on a world map with the Draw Source Map button.

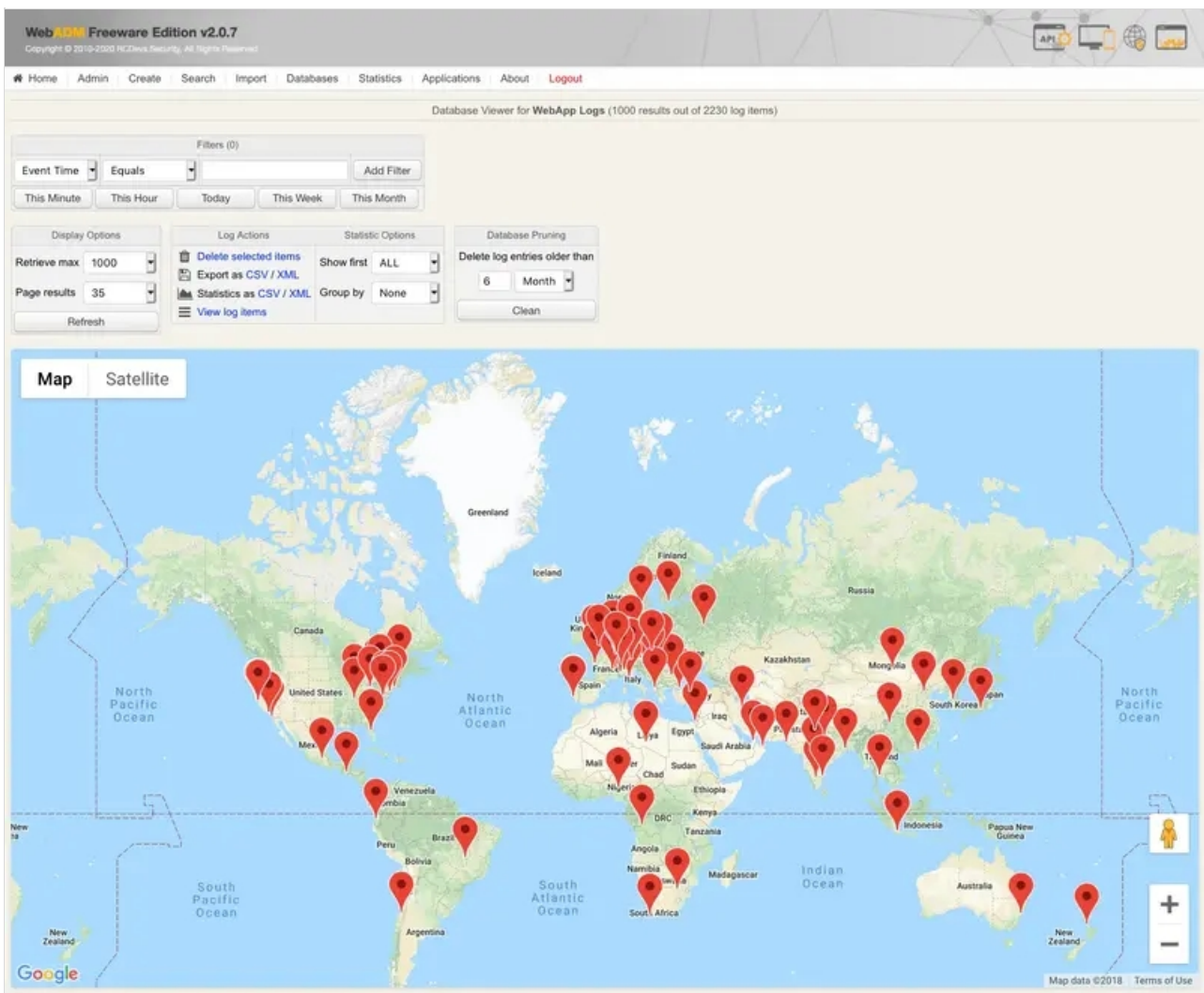


Figure 66. Source Map

## 13. Localized Messages Editor

The WebADM localized messages editor allows configuring message templates for the registered applications in different languages. There are two ways to configure WebADM applications localized messages.

1. You can review all the messages from all the applications using the messages editor accessible from the Databases menu.
2. You can go through application configurations, locate the message templates and click the Localized buttons to edit the messages in other languages.

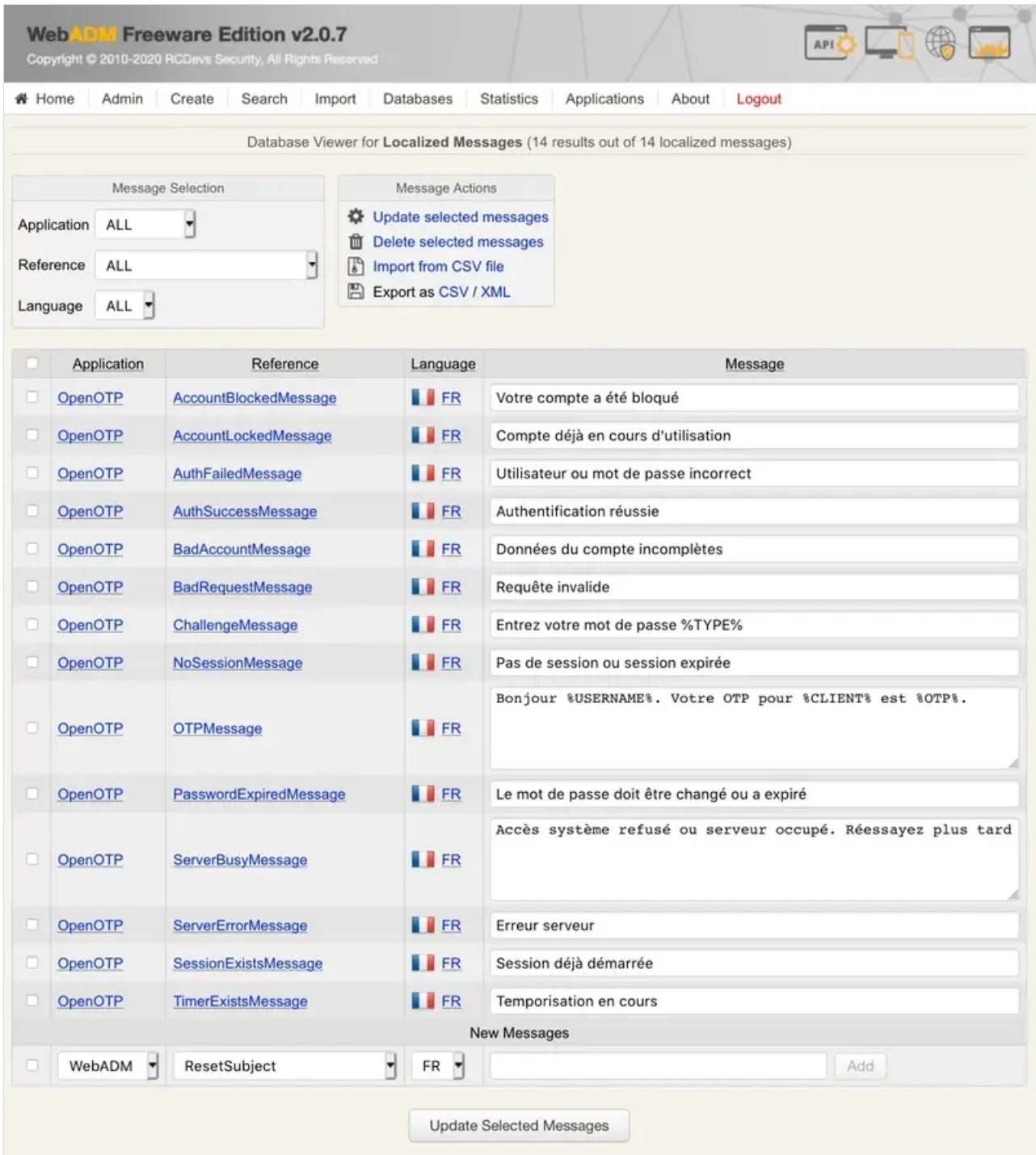


Figure 67. Localized Messages Editor

The list of supported languages is configurable in the WebADM main configuration file (`conf/webadm.conf`).

## 14. Hardware Inventory Browser

WebADM includes an inventory subsystem to be used by the registered applications like OpenOTP to store and retrieve inventoried data per-reference. The inventory is intended to ease the management of large amounts of Hardware resources like OATH OTP Tokens. For example, OpenOTP Hardware Token's registration is also possible by simply entering the Hardware Token's serial number, provided that the Token has previously been inventoried in WebADM.

The inventory is accessible through the Database WebADM menu and provides WAPI functions to let the registered applications use the inventory functionalities. The inventoried data are encrypted in the database with the same AES master key which is used to encrypt LDAP user data. Like for WebADM user data, there is per-item encryption and the inventory Type and Reference fields are used as part of the encryption process. Modifying one item's reference also invalidates the encrypted item's data.

Exported inventories are extracted with encrypted data by default. Only the inventory files provided by RCDevs and its partner Vendors are provided without the WebADM per-item AES encryption to allow the inventory import on the customer's inventory system.

The inventory provides an option to batch re-encrypt inventoried data in the event where the WebADM AES master key gets changed.

The inventory provides easy filter-based item search functionalities and allows administrators to flag items as *Valid*, *Lost*, *Broken* or *Expired*.

The screenshot shows the WebADM Freeware Edition v2.0.7 interface. The top navigation bar includes Home, Admin, Create, Search, Import, Databases, Statistics, Applications, About, and Logout. The main content area is titled "Database Viewer for Inventoried Devices (74 results out of 74 inventory items)".

Below the title, there is a "Filters (0)" section with a dropdown for "Item Type" and a search input field. Below this are buttons for "Valid", "Lost", "Broken", "Expired", "Enabled", and "Disabled".

There are two side panels: "Display Options" with "Retrieve max" set to 1000 and "Page results" set to 30, and "Inventory Actions" with options like "Delete selected items", "Scope selected items", "Re-encrypt inventory", "Check Links / Scopes", "Import from CSV file", and "Export as CSV / XML".

The main table displays the following data:

Item Type	Reference	Description	User DN	Usage Scope	Inventory Data	Active	Status
OTP Token	100588926140330	Yubikey #2101358	Link [NA]	Drop ou=RCDevs,o=Demos	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	128207798024199	Yubikey	Link [NA]	Drop ou=RCDevs,o=Demos	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	141133107972083	Yubikey #1926364	Link [NA]	Drop ou=RCDevs,o=Demos	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	141399774601770	Yubikey #1926363	Link [NA]	Drop ou=RCDevs,o=Demos	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	153572543296860	Yubikey #6954709	Link [NA]	Add [NA]	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	169869968762998	YubiKey #2573189	Link [NA]	Add [NA]	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	17492184252449	Yubikey #2573107	Link [NA]	Add [NA]	5 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	1800000398	RCDevs RC300-T6	Link [NA]	Add [NA]	6 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	1800001000	RCDevs RC300-T6	Link [NA]	Add [NA]	6 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	1800001269	RCDevs RC400-T6	Link [NA]	Add [NA]	6 Data (Hardware encryption)	<input type="checkbox"/>	Valid
OTP Token	1800001500	RCDevs RC400-T6	Link [NA]	Add [NA]	6 Data (Hardware encryption)	<input type="checkbox"/>	Valid

Figure 68. Inventory Database

## 15. Extending the LDAP Schema

WebADM relies on its own LDAP object classes and attributes. This information constitute the WebADM schema and the LDAP server using WebADM and its applications must include the WebADM schema information. The schema of your LDAP server is extended during the WebADM graphical setup (see WebADM Installation Guide for details).

When you create a MountPoint, the LDAP in the mounted LDAP server must be extended too. Once you created a MountPoint, WebADM checks if the LDAP schema is extended and proposes to add the extension if not present. This does not work with OpenLDAP where the WebADM schema file must be added to the server configuration manually. You can extend the mounted LDAP schema by editing the MountPoint object or the LDAP container where the remote LDAP is mounted. The schema extension link is included in the contextual object action box.

With Active Directory, WebADM must be connected to a Domain Controller having the schema master role for the extension to succeed.

The WebADM schema includes OIDs registered at IANA under the RCDevs' Private Enterprise Numbers 34617.

## 16. Managing Internal PKI and SSL Certificates

### 16.1 RSign Internal PKI

WebADM RSign provides features needed to automatically and immediately deliver administrator's and user's client certificates without requiring a human process. A WebADM Administrator can also issue clients/servers certificates through the WebADM GUI or Manager interface. Also, mobile certificate can be issued from OpenOTP token in order to sign a document or a transaction request for e.g.

RSign system works in client-server mode and provides a set of network remote procedure call (RPC) functions. These functions are available from a client library and are directly usable by WebADM.

The CA server component is configured to accepts or refuses client requests based on the client-provided information and rule-based filtering. It includes a configuration file (`conf/rsignd.conf`) where each client IP address must be declared and optionally given a shared access secret. RSign client requests are sent over SSL with two-way authentication.

Being a network service, the CA server component can be installed anywhere on a protected network if necessary.

### 16.2 RSign Server

It is the server component. It maintains the CA serial number, indexes issued certificates. It is multithreaded and allows processing concurrent requests. The CA is accessible to root user only but the request-processing threads run in a user-protected environment for security reasons. It uses a proprietary protocol over SSL to communicate with the client library and WebADM.

RSignd can work in proxy mode. It proxies the incoming requests to the next configured RSignd server. This can be useful when the main RSignd CA is located on a private network but should be accessed by clients on the public side through a proxy in a DMZ.

In a clustered installation, only one of the WebADM servers is running the RSignd service. The other WebADM servers will use the

main RSignd service and become PKI slaves.

## 16.3 RSign Client

RSign client is available as a client program and a shared library. RSign can be integrated into C/C++ programs. Or the client functions can be implemented in a dedicated program that can be called from other programs. WebADM uses an RSign PHP dynamic extension to implement the RSign RPC functions. The WebADM RSign client requires server configuration in the WebADM servers configuration file (`conf/servers.xml`).

The client authenticates the server through its SSL certificate.

## 16.4 Issuing Users/Administrators certificates

WebADM includes its own PKI subsystem to handle user certificates. The PKI functionalities are completely transparent and allow issuing certificates for your administrators and users using the certificate wizards.

When you create a certificate for a user, you have the following options:

- > `Certificate validity period`: Defines how long the certificate will remain valid. This period is limited to the `default_cert_validity` setting in the Rsignd signing server configuration file (`conf/rsignd.conf`).
- > `Email address`: If the user has an email address defined, you can select one email address to be part of the certificate information.
- > `Send by email`: If the user has an email address defined, WebADM can automatically send the new certificate package to the user's email address.

### Note

The generated PKCS12 certificate package is encrypted by WebADM with a random password that is not sent in the email.

- > `Certificate usage`: You can create WebADM Administration certificates for your WebADM administrators if you enabled the certificate-based login mode in the WebADM main configuration file (`conf/webadm.conf`). And you can create WebApp User certificates for your WebApp end-user if you enabled the PKI login mode for the WebApps.

### Note

Administration certificates are working with WebApps too (those configured with PKI login mode), but WebApp User certificates are not used to log in the Administrator Portal.

- > `WebApp login domain`: With WebApp User certificates, you can link the user certificate to one specific WebADM Domain. If the user is part of several domains, then only the selected domain is usable with the certificate.



**WebADM Freeware Edition v2.0.7**  
 Copyright © 2010-2020 RCDevs Security, All Rights Reserved

API | Desktop | Mobile | Security | Analytics

Home | Admin | Create | Search | Import | Databases | Statistics | Applications | About | Logout

---

New User Certificate Value(s) for cn=admin,o=Root

Certificate validity (in days):  ⓘ

Certificate export format:

*Admin certificates are used to enter Admin Portal with PKI mode.  
 User certificates are used to enter WebApps requiring certificates.*

Certificate usage:  Admin  User

User domain:

Figure 69. User Certificate Creation Form

When you issue or renew a user certificate, WebADM creates a certificate request based on the user information and calls the Rsign PKI subsystem for signing the certificate with the Internal CA. The issued public certificate is stored in the user account and can be displayed or downloaded later. The certificate and its public key are bundled into a PKCS12 encrypted package that must be provided to the user. This certificate PKCS12 package is generated once and cannot be re-downloaded later.

---

New User Certificate for cn=admin,o=Root

---

Creating private key... **Success**  
 Reading infos from LDAP user... **Success**

Certificate details:  
 - commonName: **cn=admin,o=Root**  
 - description: **ADMIN**  
 - surname: **admin**

Creating a certificate request based on the above details... **Success**  
 Calling WebADM CA for certificate request signing... **Success**  
 Checking certificate data... **Success**  
 Storing certificate in LDAP... **Success**  
 Updating OCSP cache... **Success**  
 Creating a PKCS12 package... **Success**

**Certificate installation password: 08W0Jixg**

The certificate and private key have been bundled into a PKCS12 package.  
 Click the button below to download the new certificate package.

Figure 70. User Certificate Creation

For both admin and user certificates, WebADM accepts any valid certificate it has issued provided that the login certificate used by the user to be listed in the account's public certificate list. That means that you can revoke a certificate simply by removing it from the user account, and WebADM will refuse the user login.

## 16.5 Issuing Client/Server SSL Certificates

That kind of certificates can be issued from WebADM Admin Portal or from the WebADM Manager API. Client/Server SSL certificates are stored in the SQL database configured with WebADM in `Certificates` table.

To issue Client/Server SSL Certificates, login on WebADM Admin Portal, click on `Admin` tab then click `Issue Server or Client SSL Certificate` menu.

The screenshot displays the WebADM Admin Portal interface. At the top, there is a navigation menu with items: Home, Admin, Cluster, Create, Search, Import, Databases, Statistics, Applications, About, and Logout. Below the menu, a status bar indicates: "WebADM v2.2.0 (64bit) running on server webadm1.support.rcdevs.com (192.168.4.20) in cluster mode (2 servers). Currently handling 1 connection(s)." A detailed system information section follows, listing server version details, internal server time, hardware modules, WebADM features (WebApps, WebSrvs, Manager), and RCDDevs cloud services. It also shows active LDAP, Session, and Mail servers, as well as active SQL and PKI servers. Below this, there are six dashboard cards for: User Domains (3), Client Policies (35), Access Devices (1), LDAP Mount Points (3), LDAP Option Sets (2), and Administrator Roles (1). At the bottom, there are two columns of menu items: "Licensing and Configurations" and "Runtime Actions". The "Issue Server or Client SSL Certificate" item in the Runtime Actions column is highlighted with a red rectangular box.

On the next screen you are prompted to provide information the certificate you want to issue. Provide all information you want to be part of the certificate and submit your request to Rsignd.

### Create Third-party SSL Server Certificate

You can use this form to issue a X.509 SSL certificate and private key for a third-party server or component. The certificate is generated with the provided information and signed by WebADM certificate authority.

#### Auto Confirm Mode

Enable Auto Confirm:  Yes  No i

Auto Confirm Time:  ▼

Auto Confirm App:  ▼

Auto Confirm IPs:  i

#### Main information

Server Hostname (FQDN):

Certificate Type:  ▼ i

Certificate validity (in days):  i

Private Key Password (optional):  i

#### Additional information

Alternative Name(s):  i

Organization Name:

Organizational Unit:

Country Name:  i

Locality Name:

State or Province:

Street Address:

Email Address:

Ok

Cancel

Click on **Ok** button, the certificate and the key are then issued and prompted.

Create Third-party SSL Server Certificate

Creating private key... **Success**

Certificate details:

- commonName: **testing.support.rcdevs.com**
- description: **SERVER**
- organizationName: **RCDevs Support**
- organizationalUnitName: **IT**
- countryName: **LU**
- localityName: **Belval**
- stateOrProvinceName: **Luxembourg**
- subjectAltName(s): **testing**

Creating a certificate request based on the above details... **Success**

Calling WebADM CA for certificate request signing... **Success**

Private Key (PEM format):

```
-----BEGIN PRIVATE KEY-----
MIIEVwIBADANBgkqhkiG9w0BAQEFAASCBCkwwgSlAgEAAoIBAQDezjueHrWi0U81
pSSxm/MSCqrdVlVRDwml+mRCQkntqzNo8DnAC9LwC46MFnFFVrpEURR1A/FFFNp
Y/wx2mpKMLfMVX4BwJ9V/0oCbODXVlCTiRHGmrAgWxrvYhEY0GKMTvOuHAYgoO/
0B3PMY+CxQzPFs/rmPdNra+EI3hKgc/0+tJo4fdvqYVDzaWd9X0D0uYogSulv5nT
24CsSmzuIDyFE0aq0V42lu8jvz+/EUKVo3KokQaIX3nro0XiJFNSvpNgsVy067Fq
CTuUHjsioTBFYVhu9Nj332wHUI2Qjszlj2dlFwsfF8+nHNdW89MbZ6M4UBakfH7m
J5jvu0qTAgMBAECGgEATTGCskvNfnWgTdnDls2DL7MGc90hvpI5EkoRhsVbIRu
7QhZiy+LM2GIpzz3TiC4NEgZeS4ef4a8V6018jmmoJGrQLjv22vRqpgSATU+YZs
MFGDIffPHK8UitLtzVPeNzTLldm7ND7OI+wYmVe/veoMeqjaur6c61DbDKl0Jn3
lQbxJdETvkXmWoEWmljebCageD0rZ7o4eMVmUsNntc+Zpfc8Xxln3k4EM/t+j14
dyXvqHBF2pFqHtSGCVpQZ+rPpv10pr7dJUUJg+iIVtsVx0t5AVLoc+Eu+AllDtQ
PvD8lZEfmk3ZIWOGTExc3J2oD74EIsT36MzWgDRwQKbQD17Tl6fu0ICYj8C8EJ
fga5bjFsEyoJDJevqJ2dXGNsXCSEbYkH0R/sFVStmq7DMAFQEQK5J/HiCiQkgyhU
OS4FGw4+PaB6mMk03oKoF/F479eEYyZlhNwoZmlIH8wE5Y+e7aEO8vzOVneJvTsh
uJdEDLbCccGhdL6nSYl7QJA2wwKBgQDn7o/eY2GU0K0ITWMSumzjrk3hVi0zncf7
Mh6ePw1idQU+Tlg6vsysMgDYovePNQDj+q6U5svK/R8VnkCz+xklrxtXA7XNDvfk
v1Ptnculua7bAvp713uYlU5iSlfvIHj+0RzNdevJlAJKZMds7NwAHGUc8RAM5tt
dw+vrS9/8QKbGQGMFscrAVKTWmhngHGlalIoNr2zNs09NoPz0Gxi6f0iJKkb+BF
kx6Hs1uMh1A49AbRWbSASis3S5ThgaLbWfFA3V0D0ew4+PR8E2z4SoNBhhu9Me0
-----
```

Certificate (PEM format):

```
-----BEGIN CERTIFICATE-----
MIIEEDCAxyGwIBAgICAI4wDQYJKoZIhvcNAQELBQAwVzEQMA4GALUEAwHUm9v
dC8BDQTEAMBgGALUECwwRUKNEZXXzIFNlcnZpY2UgQ0ExGjAYBgNVBAoMEVJDRGV2
cyBTZXJ2aWw1IFNBMCwCQYDVQQGEwJMVTEpMA0GA1UEBwwGQmVudmFMRMRwEQYDVQ
IiEAMjM5NTNTNAmIGQMSMwIQYDVQQDBp0ZXN0aW5nLnNlcHJvcnQumNkZXXzLmNv
bTEPMA0GA1UEGgwU0VSVkVSMRgwFgYDVQQKDA9SQ0R1dnMgU3VvcG9ydCAXCzAJ
BgNVBAsMAk1UMQswCQYDVQGEwJMVTEPMA0GA1UEBwwGQmVudmFMRMRwEQYDVQ
IDApMdxlbWJvdXJnMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs47
nh61otFPNaUksZvzEgq3VZb0Q1ppfpkQkJ7ap8zaPA5wAvS8Au0jBzxRVA6RFP
EZQPXRZzawP8MdpqSjJxzFV+AcCfVf9KAmzq1lZQk4kR4BpqwK1l672IRGNBiJem
zrhwG1Kdv9AdxTGPgsUGTxbP65j3Ta2vhCN4SoHP9PrSaOH3VamFQ821nfv9Azrm
KIErbp+Z09uArEps7ia8hRdmgtPeNpbvI72fvxPCLaNyjpEGiP9566NMSixTUR6T
YLFctOuxagk7lB47IqEwrWFYbvTsd49sB1CNkI7M5Y9nZRVRHxfPpxzXvPTG8+j
OFAWpHx+5ieY77tKkwIDAQABo4HPMIHMMBIGALUdEQQLMAMCB3Rlc3RpbmcwCwYD
VR0PBAQDAgXgMBMGA1UdJQQMMAoGCCsGAQUFBwMBMGkGCCsGAQUFBwEBBF0wWzAl
BggRBgEFBQcAwAYYZaHR0cDovLzE5Mi4xNjguNC4zMS9vY3NwLzAyBggrBgEFBQc
wAoYMAHR0cDovLzE5Mi4xNjguNC4zMS9jYWNlcnQvP2ZvcmlhdDlkZlXwKQYDVR0f
BCIwIDAoByggOYYaHR0cDovLzE5Mi4xNjguNC4zMS9jcmVvMA0GCsGCSIB3DQEB
CwUAA4IBAQB76ixz2ZCrs2tnWbf/6VMSNnotCCzhXgOEHRc7YE2+hi9UCT/KFHY
ad5f9k8t80a1e0e0u/i773x2b10qxx0RaaMDhcvTARCNMv5Xf6SLlTv
-----
```

You can download certificate and its associated key and use it wherever you need. For client certificate, you just need to set **Certificate Type** setting to **Client** instead of **Server** by default :

You can use this form to issue a X.509 SSL certificate and private key for a third-party server or component. The certificate is generated with the provided information and signed by WebADM certificate authority.

---

**Auto Confirm Mode**

Enable Auto Confirm:  Yes  No i

Auto Confirm Time:  v

Auto Confirm App:  v

Auto Confirm IPs:  i

---

**Main information**

Client Name or Description:

Certificate Type:  i

Restricted Application:  i

Certificate validity (in days):  i

Private Key Password (optional):  i

---

**Additional information**

Organization Name:

Organizational Unit:

Country Name:  i

Locality Name:

State or Province:

Street Address:

Email Address:


If you set the setting **Restricted Applications**, then the issued client certificate can be used only with the targeted application (e.g: OpenOTP).

## 16.6 Mobile SSL Certificates

Mobile certificates are used for document Signin purposes with OpenOTP Signature APIs. Please refer to [OpenOTP Signature](#) for more information on that part.

A mobile certificate is a **User** certificate but stored in the SQL database and not on the **User** account like regular user

certificates. The **Certificate type** in the SQL database is **Mobile**.

<input type="checkbox"/>	Cert.Type	Serial	Common Name	Application	Creation Time	Expiration Time	Last Use	Host IP	Certificate	Enabled	Renew
<input type="checkbox"/>	MOBILE	144	test	OpenOTP	2023-01-23 17:16:54...	2023-02-22 17:21:54...	2023-01-23 17:21:56...	192.168.4.20	Valid  		

You can revoke it if needed. Have a look on the next section to revoke it.






## 16.7 Manage issued Certificates

You can revoke an issued certificate whenever you want.

For **User Certificates**, you need to go on the user account and delete it from the user account directly. It is then directly revoked and unusable.






For **Client/Server/Mobile SSL Certificates**, you need to go on **WebADM GUI** > **Databases** > **SQL Data Tables** > **Client, Server & Mobile Certificates** menu.

### SQL Log Tables

-  [Administrator Logs](#)  
Admin Portal logs (admin audit)
-  [Manager Logs](#)  
Manager Interface logs (admin audit)
-  [WebApp Logs](#)  
Web Application logs (user audit)
-  [WebSrv Logs](#)  
Web Service logs (user audit)
-  [Alert Logs](#)  
System Alerts from applications

---


### SQL Data Tables

-  [Localized Messages](#)  
Message translations for applications and services
-  [Inventoried Devices](#)  
OpenOTP hardware tokens and SpnKey PIV keys
-  [Recorded Sessions & Transactions](#)  
Transaction records and SpanKey sessions' audit
-  [Physical Access & Mobile Badging](#)  
Dashboard with badging records and presence reports
-  [Client, Server and Mobile Certificates](#)  
Provides review and revocation for services your certificates

---

### System Log Files

You are going to find here, all issued certificates after the WebADM setup has been performed. Found below, the certificate I issued in that example.

<input type="checkbox"/>	Cert Type	Serial	Common Name	Application	Creation Time	Expiration Time	Last Use	Host IP	Certificate	Enabled	Renew
<input type="checkbox"/>	SERVER	142	testing.support.rcde...	[NA]	2023-01-23 16:39:53...	2024-01-23 16:39:53...	2023-01-23 16:39:53...	[NA]	Valid 	<input checked="" type="checkbox"/>	<input type="checkbox"/>

If I want to revoke it from CRLs and OCSP endpoints, I just need to turn off the **Enabled** button.

<input type="checkbox"/>	Cert Type	Serial	Common Name	Application	Creation Time	Expiration Time	Last Use	Host IP	Certificate	Enabled	Renew
<input type="checkbox"/>	SERVER	142	testing.support.rcde...	[NA]	2023-01-23 16:39:53...	2024-01-23 16:39:53...	2023-01-23 16:39:53...	[NA]	Revoked 	<input type="checkbox"/>	<input type="checkbox"/>

Certificate is now revoked. I have to keep that entry into the SQL database in order to keep it as **revoked**. If I removed it, then the OCSP and CRL endpoints will not be able to provide the status of that certificate.

The **Renew** button is working only for RCDevs products (e.g: Radiusd, Spankey, WAProxy...). To disable the certificate auto-renewal, you just need to turn off the **Renew** button. If auto-renewal is enabled but not supported on the client application side, then it's not a big deal. That setting just flag a certificate entry as **Renewable** for RCDevs products which support the automatic renewal.

<input type="checkbox"/>	Cert Type	Serial	Common Name	Application	Creation Time	Expiration Time	Last Use	Host IP	Certificate	Enabled	Renew
<input type="checkbox"/>	SERVER	142	testing.support.rcde...	[NA]	2023-01-23 16:39:53....	2024-01-23 16:39:53....	2023-01-23 16:39:53....	[NA]	Revoked ⓘ	<input type="checkbox"/>	<input type="checkbox"/>

## 16.8 CRLs and OCSP Endpoints

WebADM infrastructure comes up with OCSP (Online Certificate Status Protocol) and CRLs (Certificate Revocation Lists) which are two methods used to check the revocation status of digital certificates. OCSP allows for real-time certificate status checking by querying a certificate authority's OCSP server, while CRLs provide a list of revoked certificates that is periodically published by the CA. Both methods are used to verify that a certificate is still valid and has not been revoked. OCSP is considered to be more secure and efficient, as it allows for real-time checking, while CRLs rely on a regularly-updated list which can be out of date. These 2 endpoints are local URLs which are automatically published on WAProxy or another reverse proxy if WAProxy or a RP is configured with your WebADM infrastructure.

The default URLs are :

- > `http://webadm_server/ocsp/`
- > `http://webadm_server/crl/`

When WAProxy/Reverse Proxy servers are configured, the URLs are :

- > `http://waproxy_or_reverse_proxy_public_url/ocsp/`
- > `http://waproxy_or_reverse_proxy_public_url/crl/`

OCSP and CRL endpoints are automatically added to all issued certificates. You can verify it by reading the content of an issued certificate through WebADM GUI or with OpenSSL commands.

### 16.8.1 OCSP Check

The OpenSSL command can be utilized to verify certificate revocation status using the OCSP service:

```
openssl ocsp -issuer ca.crt -cert johndoe.crt -text -url http://webadm1.support.rcdevs.com/ocsp/ -header "HOST"="webadm1.support.rcdevs.com"
```

#### 16.8.1.1 Valid Certificate

For a valid certificate, the aforementioned OpenSSL command will yield the following result (status: god):

OCSP Request Data:



Version: 1 (0x0)

Requestor List:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: B1898F8D6DE91859F6CA87B4EA18A70E4231A3A9

Issuer Key Hash: BAC6DDBC32CE57DECE3FC9ED4E8D0867BFA9F08C

Serial Number: AD9DBEC023CB7C50047DDF178164097F

Request Extensions:

OCSP Nonce:

04101E8F6CE17FBEB7F5F3B07A4D3A0F0811

OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: CN = RCDevs Support CA, OU = IT, O = RCDevs Support SA, C = LU

Produced At: Dec 13 14:32:53 2023 GMT

Responses:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: B1898F8D6DE91859F6CA87B4EA18A70E4231A3A9

Issuer Key Hash: BAC6DDBC32CE57DECE3FC9ED4E8D0867BFA9F08C

Serial Number: AD9DBEC023CB7C50047DDF178164097F

Cert Status: good

This Update: Dec 13 14:32:53 2023 GMT

Response Extensions:

OCSP Nonce:

04101E8F6CE17FBEB7F5F3B07A4D3A0F0811

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

6e:c1:8c:a3:03:ba:4b:4c:4f:d0:73:92:9f:8e:c8:9e:2b:d2:  
25:01:e2:f5:15:da:7e:a7:0e:52:66:39:18:d0:86:be:f7:38:  
d1:09:bc:a4:2b:c7:e6:bc:96:23:a0:10:72:bf:45:b3:1e:78:  
80:2a:ea:6c:bd:22:bd:28:77:9b:c1:c9:de:5e:9e:f2:6c:d6:  
7f:65:a8:15:7c:28:97:a3:dd:4b:3c:d3:79:03:09:ab:c7:90:  
2f:4f:de:f6:f4:05:7f:69:5d:80:20:42:6c:0e:cc:b9:ea:29:  
7f:9f:b9:27:90:27:10:35:35:7d:2c:83:be:fb:0d:a8:4a:79:  
0a:5d:64:dd:ed:1b:a3:c1:49:0c:64:8b:c8:6e:4a:54:f0:6c:  
16:0c:4f:78:12:fe:df:5f:e8:42:eb:97:66:7b:91:4c:0e:51:  
59:7b:13:5b:26:38:a7:10:ca:19:0d:cc:43:20:82:5a:8f:ec:  
40:c2:e6:f7:a0:38:1d:5c:44:2c:62:3b:3e:2e:c3:e6:90:cd:  
d6:8e:e5:c6:b5:04:10:ca:b9:3f:7e:cb:54:fb:30:b9:ec:d0:  
b3:7c:42:79:6f:3c:83:ce:23:9e:9f:45:0f:66:f1:f5:be:ab:  
af:4b:b3:4d:ec:c9:d8:9c:30:8d:42:87:c9:b7:55:3b:d8:2a:  
c1:5a:7a:27:77:45:b0:a4:de:30:a8:cc:62:d2:50:35:d7:2d:  
bd:93:66:a4:d5:cd:62:a8:f1:ba:d0:1f:1e:c3:df:07:81:3e:  
fd:8b:7b:1c:a5:6b:44:df:7f:eb:71:26:70:48:85:a9:37:29:  
ff:23:dd:f8:fa:65:59:4a:9c:ea:f9:7c:88:8d:32:c7:75:2e:  
f9:b5:66:db:1c:b9:95:67:89:86:bf:36:18:86:ba:d4:7c:d6:  
fa:17:ac:ac:82:ba:74:35:42:35:0f:0a:ef:cf:07:9f:d6:8e:

```
1a:17:ac:ac:02:0e:74:55:42:55:01:0a:01:07:51:00:0e:
6c:93:eb:68:11:4b:5a:7c:2f:1a:ec:90:fb:b6:90:2b:12:28:
a8:87:f8:1d:95:ab:b5:d6:e0:8a:a4:ab:c6:2b:7e:7f:9d:14:
f3:24:ae:46:eb:af:ac:8f:0d:43:a4:f5:3c:15:34:8e:74:9d:
05:a9:11:37:76:f5:91:00:b1:e6:0f:8e:40:ce:38:e2:7e:8f:
0f:ee:1a:42:53:77:ac:63:4d:00:5f:74:d1:bb:39:e8:be:93:
b1:37:28:04:cd:ea:1a:4e:8a:ba:05:ea:a6:bc:f4:3c:54:a3:
72:18:98:ad:3b:e9:74:a2:a6:d6:26:cc:e9:00:85:d2:18:b2:
f0:97:3c:c6:c3:5b:92:3b:11:dd:0e:c6:1c:db:b4:da:65:98:
20:a0:ed:65:20:3e:f5:ec
```

Response verify OK

johndoe.crt: good

This Update: Dec 13 14:32:53 2023 GMT

Below, the WebADM logs regarding the previous request:

```
[2023-12-13 15:32:53] [192.168.3.205:63375] New OCSP request for serial:
230775502758290284840807186191261895039
[2023-12-13 15:32:53] [192.168.3.205:63375] > Issuer Hash:
b1898f8d6de91859f6ca87b4ea18a70e4231a3a9 (SHA1)
[2023-12-13 15:32:53] [192.168.3.205:63375] Returning OCSP response 'Good'
```

### 16.8.1.2 Revoked Certificate

A certificate is deemed revoked for the following reasons:

- > **User certificate:** Certificate not existing on the user account.
- > **Client certificate:** Certificate marked as **Revoked** in the SQL database or removed from the SQL database.
- > **Server certificate:** Certificate marked as **Revoked** in the SQL database or removed from the SQL database.
- > **Mobile certificate:** Certificate marked as **Revoked** in the SQL database or removed from the SQL database.

#### Note

Mobile certificates used for document signing are revoked only under two conditions: if they are labeled as “Revoked” in the SQL database or if they have expired. If a certificate is still valid in terms of its expiration date but has been removed from the SQL database, it can be used during its validity period and will be automatically re-added to the SQL database. To render a mobile certificate unusable, it must be retained in the SQL database and marked as **Revoked**.

For a revoked certificate the OpenSSL command previously provided will return the status revoked and the revocation time:

```
OCSP Request Data:
Version: 1 (0x0)
```

Requestor List:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: 66B72E282CBB66675F45363B7B9667AB5F1DC68D

Issuer Key Hash: 04405B3B546C1F93E5CF15C033D21C51A17565A3

Serial Number: 80502E9B6C05534A965C104D6E182743

Request Extensions:

OCSP Nonce:

041029DD3F7A25F4286C09099EE96A358860

OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: CN = WebADM CA #113f15bb, O = RCDevs Testing

Produced At: Dec 14 13:12:30 2023 GMT

Responses:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: 66B72E282CBB66675F45363B7B9667AB5F1DC68D

Issuer Key Hash: 04405B3B546C1F93E5CF15C033D21C51A17565A3

Serial Number: 80502E9B6C05534A965C104D6E182743

Cert Status: revoked

Revocation Time: Dec 14 13:12:30 2023 GMT

This Update: Dec 14 13:12:30 2023 GMT

Response Extensions:

OCSP Nonce:

041029DD3F7A25F4286C09099EE96A358860

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

79:ec:1a:84:01:67:c4:36:27:f6:26:54:c3:b6:7f:01:78:90:  
7e:3a:f0:3b:14:b5:a5:18:32:8a:66:22:5a:e2:b4:ab:85:d6:  
3d:15:68:22:b5:b1:c9:26:59:ab:d2:45:e4:07:ea:16:a4:97:  
bb:3f:27:2a:72:73:41:da:c0:74:f8:60:d3:e7:3c:ce:7c:72:  
d0:54:2d:99:61:dc:07:2e:04:a7:d3:fe:13:7b:73:9c:14:92:  
6c:ad:b9:e2:a5:3a:fb:88:db:25:97:92:66:44:47:41:7e:2c:  
4c:00:df:e2:38:d5:7a:37:9a:82:49:ab:48:53:03:41:0f:25:  
dd:35:93:54:d6:d3:da:21:a4:35:cb:d2:92:7e:a0:43:75:7b:  
6e:85:a8:1d:88:2c:2e:0f:e7:3c:0e:f7:6c:38:8e:e2:02:82:  
a7:12:37:de:75:92:c3:8d:4a:a2:b1:dc:ce:06:70:99:6d:ee:  
73:1d:1b:ef:6f:23:4b:68:28:13:c8:bf:63:ab:c2:25:d4:ba:  
0e:03:f9:62:c9:15:3e:d5:1e:ba:09:44:cf:ab:c7:9c:75:3a:  
fe:23:fd:43:bf:b4:eb:15:0f:e0:20:ca:ba:69:c3:e9:c3:0c:  
5c:d7:51:ea:4f:d9:69:3c:e4:73:be:e7:f1:79:4d:ac:25:88:  
a1:33:58:3d:51:c5:08:df:41:00:b6:89:11:b6:68:0c:23:d9:  
73:b5:ea:b9:7c:a8:87:70:cd:1a:10:af:ec:04:2d:cf:09:72:  
94:fd:c3:16:c1:4f:c7:56:a0:52:99:65:9c:36:12:1f:3d:82:  
78:27:fd:ec:8f:7e:04:6a:80:b7:4c:70:71:0a:b2:16:d0:16:  
f9:23:05:fa:de:e9:71:a9:62:49:15:3a:a7:c0:69:93:62:da:

```
c1:f1:1c:50:fd:22:d1:02:47:ef:3e:21:39:18:cf:11:75:54:
2e:d0:30:83:13:33:83:2c:cd:9a:c4:a7:77:95:0d:aa:7d:ad:
93:ae:6e:b9:39:b0:34:b8:cf:8b:c9:1c:2b:86:1d:f9:0d:ae:
c5:b3:b5:b3:6e:84:6e:14:bc:3e:c4:2b:fe:6e:23:76:9e:28:
38:2c:fa:5a:a1:6a:1d:f5:82:95:8d:8a:85:c4:f8:28:dc:39:
b3:52:2c:26:43:0f:e9:c0:21:ad:76:9a:8a:9b:b9:c3:d6:1f:
bf:57:69:a5:0f:aa:0a:1d:14:1f:a5:09:83:04:72:be:9b:40:
fd:84:c8:3a:85:a4:bd:ad:bd:16:8e:03:bc:eb:17:12:9a:57:
a9:1b:07:6f:91:e0:36:33:e5:4c:d9:9b:bb:9a:c5:60:f5:ad:
f0:b0:3a:65:e0:0f:00:8c
```

Response verify OK

yoann\_dev\_ok.crt: revoked

This Update: Dec 14 13:12:30 2023 GMT

Revocation Time: Dec 14 13:28:22 2023 GMT

You can find OSCP requests logs in `/opt/webadm/logs/webadm.log`.

```
[2023-12-14 14:12:30] [192.168.3.205:57252] New OCSP request for serial:
170557512513789794651896604926594787139
[2023-12-14 14:12:30] [192.168.3.205:57252] > Issuer Hash:
66b72e282cbb66675f45363b7b9667ab5f1dc68d (SHA1)
[2023-12-14 14:12:30] [192.168.3.205:57252] Returning OCSP response 'Revoked'
```

### *16.8.1.3 Certificate expired*

For an expired certificate (user, client, server or mobile) available on the user account or in the SQL database (not flagged as revoked), the OpenSSL command previously provided will return the status `unknown`:

#### OCSP Request Data:

Version: 1 (0x0)

Requestor List:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: B1898F8D6DE91859F6CA87B4EA18A70E4231A3A9

Issuer Key Hash: BAC6DDBC32CE57DECE3FC9ED4E8D0867BFA9F08C

Serial Number: 8A4F601706022C8139B0F4D9A7656BFC

Request Extensions:

OCSP Nonce:

04102F59FCB9622706FE66F3168E90E9DCE8

#### OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: CN = RCDevs Support CA, OU = IT, O = RCDevs Support SA, C = LU

Produced At: Dec 14 14:54:31 2023 GMT

Responses:

Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: B1898F8D6DE91859F6CA87B4EA18A70E4231A3A9

Issuer Key Hash: BAC6DDBC32CE57DECE3FC9ED4E8D0867BFA9F08C

Serial Number: 8A4F601706022C8139B0F4D9A7656BFC

Cert Status: unknown

This Update: Dec 14 14:54:31 2023 GMT

Response Extensions:

OCSP Nonce:

04102F59FCB9622706FE66F3168E90E9DCE8

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

53:5d:de:a6:76:12:8d:b8:53:73:3a:41:35:39:23:da:9e:13:  
fb:e5:b5:1f:62:97:9f:f3:31:6a:98:a6:5d:7a:71:e0:54:23:  
1b:07:03:0a:23:3f:83:a8:26:95:b0:ba:e4:d2:a3:f0:69:39:  
2b:99:e5:aa:9c:13:f0:1c:a8:60:fa:1a:31:1f:24:da:dd:97:  
d8:ac:60:57:ef:77:c5:0d:6c:d5:50:e8:13:0f:8a:3d:eb:e7:  
65:ac:89:93:97:d0:e3:f6:84:c4:45:7c:5e:88:05:fa:79:76:  
f8:78:90:86:f0:02:b4:e0:c4:6f:54:0c:b0:c8:95:40:1a:b3:  
46:1a:b8:b4:48:20:99:4b:80:cb:c6:3a:a6:78:cf:6c:d4:ef:  
83:fd:31:51:57:44:40:39:5e:a2:36:fd:10:b5:d3:c1:07:dd:  
72:c9:7b:88:be:40:ca:07:22:b0:37:b1:2b:59:e0:47:71:df:  
a8:eb:3e:19:87:f3:99:e5:bd:9f:7e:85:c7:bc:2a:14:13:44:  
56:25:f0:d8:6c:a3:03:52:8f:c2:d1:e0:6e:07:64:70:3f:e8:  
56:76:f0:91:7e:9b:3f:78:6d:28:41:6d:8d:cd:50:b5:7e:7a:  
f3:fd:1c:4a:85:59:db:74:df:92:15:a3:ba:8f:cf:14:4a:e2:  
12:69:f2:f6:96:1a:51:21:fa:51:f2:d9:09:8a:ae:cd:24:f3:  
73:fe:79:a2:26:b9:da:66:b1:46:26:78:69:d9:9b:91:d5:00:  
e2:cd:66:14:dd:1b:d6:a4:61:39:d1:48:71:01:33:50:ac:38:  
e5:e8:28:f2:f2:98:a6:73:bf:b6:a1:7b:a9:7c:da:be:15:40:  
3b:e0:d5:39:a1:43:58:4d:49:5c:9b:b4:b6:a1:ff:48:75:c2:  
58:84:56:c3:ef:0e:50:61:f3:08:20:0f:d1:dc:c3:8c:77:ad:  
b7:84:8a:1e:88:9d:0b:a6:ea:f4:d8:ec:d7:e3:3a:ea:28:6c:  
c1:6b:85:68:c2:5b:75:0a:d0:26:d7:ac:6d:32:be:89:5f:17:  
86:0e:46:6c:b1:d8:7d:5b:b0:af:d4:95:a3:b5:c8:4f:8f:a2:  
54:9d:30:a8:db:a0:18:78:05:4f:f9:9c:0a:c6:e0:75:42:d4:  
a7:26:d1:8b:3b:39:a3:21:87:21:90:db:68:c1:33:9b:33:f4:  
d6:fa:5c:d0:cf:d5:1a:fc:38:b4:ad:04:f6:95:9a:c1:23:f6:  
3a:b3:d7:4c:32:4f:28:42:29:78:ed:a5:0d:41:d0:ea:bc:f8:  
cd:91:55:af:f0:45:12:af:46:8e:9b:7f:6e:3e:92:6b:cc:8e:  
34:bf:eb:4e:29:6a:e9:46

Response verify OK

webadm1\_valery\_expired.crt: unknown

This Update: Dec 14 14:54:31 2023 GMT

You can find OSCP requests logs in `/opt/webadm/logs/webadm.log`.

```
[2023-12-14 15:59:18] [192.168.3.205:63898] New OCSP request for serial:
183845603805571868310299370231666404348
[2023-12-14 15:59:18] [192.168.3.205:63898] > Issuer Hash:
b1898f8d6de91859f6ca87b4ea18a70e4231a3a9 (SHA1)
[2023-12-14 15:59:18] [192.168.3.205:63898] Returning OCSP response 'Unknown'
```

#### 16.8.1.4 Invalid issuer (Wrong CA)

OCSP request for a certificate not issued by WebADM internal PKI will also return the `unknown` status:

##### OCSP Request Data:

Version: 1 (0x0)

##### Requestor List:

##### Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: FEB81015ABD71BC178CBAB41E58A1AEF08454527

Issuer Key Hash: 04405B3B546C1F93E5CF15C033D21C51A17565A3

Serial Number: 01EE

##### Request Extensions:

##### OCSP Nonce:

0410CEE701CE0DF0A9E2101575F81D4FF751

##### OCSP Response Data:

OCSP Response Status: successful (0x0)

Response Type: Basic OCSP Response

Version: 1 (0x0)

Responder Id: CN = WebADM CA #113f15bb, O = RCDevs Testing

Produced At: Dec 14 13:10:26 2023 GMT

##### Responses:

##### Certificate ID:

Hash Algorithm: sha1

Issuer Name Hash: FEB81015ABD71BC178CBAB41E58A1AEF08454527

Issuer Key Hash: 04405B3B546C1F93E5CF15C033D21C51A17565A3

Serial Number: 01EE

Cert Status: unknown

This Update: Dec 14 13:10:26 2023 GMT

##### Response Extensions:

##### OCSP Nonce:

0410CEE701CE0DF0A9E2101575F81D4FF751

Signature Algorithm: sha256WithRSAEncryption

##### Signature Value:

5a:92:13:76:ae:90:56:1c:a0:b3:0d:df:27:6f:7c:b0:27:a0:

1d:2a:a1:a4:27:2b:80:c4:79:4b:09:00:8d:3d:98:73:d8:7d:

04:64:00:2b:11:f2:ce:89:8c:10:02:cf:61:c2:92:ee:19:89:

8f:6f:d5:1a:dd:40:13:11:85:c5:36:ea:23:51:85:f0:b5:fc:

```
16:2a:eb:1b:5f:50:86:eb:0c:26:14:6e:44:ff:f7:95:47:3a:
19:99:8f:6b:1f:64:12:29:69:74:5b:88:61:0d:c9:b0:13:4c:
61:e2:d7:eb:51:b5:13:d7:ac:51:89:f4:ad:67:b1:ee:bc:e1:
cf:4b:25:f1:48:e7:ca:a3:55:50:ad:e7:8c:46:c3:f5:61:8a:
92:dc:92:0f:b3:ca:25:54:18:eb:1a:bb:bd:14:64:c2:6a:5e:
6e:14:d9:00:d1:70:bd:b2:79:eb:55:35:33:ce:39:83:91:63:
4e:4b:1d:82:f6:a9:3b:3b:19:40:85:b0:32:42:7d:9a:80:f5:
72:ba:bb:c3:7a:d0:1b:e7:44:40:01:cc:71:fb:f1:a4:28:b0:
80:f4:82:bd:92:61:c8:9e:35:9a:ca:5a:7b:ca:5c:15:be:35:
26:58:93:cc:3a:f7:5f:2b:d5:dd:01:97:6e:2b:9c:67:06:41:
7a:0a:e5:c0:7b:27:03:90:f8:c9:2c:6d:1a:8d:e8:ef:0b:a3:
75:66:c9:2f:c9:08:2d:5f:c2:67:ea:77:2d:ed:3e:1c:46:09:
96:47:fd:d5:75:a9:d2:4a:cd:e6:52:8c:28:ef:cb:ea:5c:71:
29:ea:81:e5:dc:a1:b7:84:05:50:80:1b:93:fe:be:18:8c:6b:
d9:70:82:5e:0d:ec:2a:1b:5a:ca:be:0d:e2:fc:3f:14:2b:8d:
dc:bf:ae:4c:08:9e:51:01:e5:87:0d:2e:56:b8:c1:be:f1:24:
f7:ac:fc:cf:6b:ff:f3:4e:76:48:9c:53:c4:01:5b:b2:68:e7:
d9:33:c3:96:a7:f7:aa:a9:f8:e7:74:03:85:39:c2:51:06:ca:
eb:a8:86:a7:5b:03:da:b9:c2:05:52:2b:26:ee:b2:ad:bf:45:
b5:5a:e7:82:23:9e:97:2e:0b:64:f5:e0:14:60:dc:84:16:2d:
30:f7:55:a3:d2:57:c2:1d:b9:6d:e9:16:39:36:bf:ed:c2:15:
81:70:3e:bc:8a:e1:1f:a8:fc:c3:0c:2c:a9:24:48:74:55:13:
b5:1c:52:7c:f3:35:98:d0:16:3a:85:9f:8b:e0:d8:78:d4:01:
f5:ed:22:13:fa:d7:2c:70:dd:c5:8f:d4:3b:6e:77:da:d2:2e:
3d:b3:ee:69:0c:6d:3a:5c
```

Response verify OK

wrong\_CA.crt: unknown

This Update: Dec 14 13:10:26 2023 GMT

### 16.8.2 CRL Check

The CRL endpoint can exclusively be employed for checking the revocation status of SQL-stored certificates. When using the CRL method, it is crucial to retain all revoked or expired certificates to construct the CRL with their serials.

OpenSSL can be employed to verify certificate revocation by utilizing a retrieved CRL.

The following command enables you to download the CRL from WebADM in DER format:

```
wget http://webadm1.support.rcdevs.com/crl -q -O webadm1.crl.der
```

The following command will read the CRL file in DER format and furnish information regarding certificate revocation based on serial numbers:

```
openssl crl -inform DER -in webadm1.crl.der
```

Certificate Revocation List (CRL):

Version 2 (0x1)

version 2 (0x1)

Signature Algorithm: sha256WithRSAEncryption

Issuer: CN=RCDevs Support CA, OU=IT, O=RCDevs Support SA, C=LU

Last Update: Dec 14 15:45:29 2023 GMT

Next Update: Jan 13 15:45:29 2024 GMT

CRL extensions:

X509v3 Authority Key Identifier:

keyid:BA:C6:DD:BC:32:CE:57:DE:CE:3F:C9:ED:4E:8D:08:67:BF:A9:F0:8C

DirName:/CN=RCDevs Support CA/OU=IT/O=RCDevs Support SA/C=LU

serial:32:49:B4:20:D8:25:78:93:95:5A:B1:87:AD:8C:13:43:85:A1:AD:03

X509v3 CRL Number:

1

Revoked Certificates:

Serial Number: BA3F5DD65B864EFA98B0F4484E98471E

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: D4EA92954DEFA9376B9FE4158740586F

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: B09E8A84E19614D40B2B49235BE0D41E

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: D105426C2604181853CE8CAE016A3D19

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

71:28:a5:0d:a0:51:73:26:62:2a:05:8e:cc:13:b6:43:7c:dc:  
46:0b:81:08:cc:16:39:4a:96:af:07:d8:ad:45:db:5a:d1:3c:  
2e:65:53:07:ff:1f:45:d1:9c:e8:a0:e3:9c:10:98:3b:cd:1c:  
91:90:f1:d1:60:79:53:39:4a:d7:49:d0:ab:5c:b9:61:1b:2e:  
2a:6d:1b:43:c9:9e:7b:95:86:05:c0:46:b9:ed:da:4d:dd:bd:  
b6:b4:78:1e:7f:1e:6d:5d:1f:15:2e:dd:bb:e7:13:96:c1:99:  
01:6e:a1:d1:5a:48:e7:c1:ab:11:b7:eb:14:24:ea:77:c9:81:  
ea:cc:84:86:20:d8:7f:f5:a5:0e:57:fb:21:ee:ed:e2:53:97:  
2c:47:09:ac:59:10:8a:25:1c:29:bf:60:a9:4d:3e:e4:8f:aa:  
7d:ad:87:d6:9f:73:30:23:39:51:6e:3e:dc:25:60:38:f2:df:  
bb:29:b2:f3:28:3e:e6:24:dc:d7:87:e0:b4:94:2d:2e:87:0c:  
3c:8e:a9:c1:95:03:70:ee:13:57:8c:93:a5:13:31:b7:4e:43:  
71:0d:3c:a6:de:9f:31:70:8f:e3:88:f5:59:d6:ff:21:47:4c:  
2e:1f:64:f8:b4:a8:d8:02:49:74:24:54:d8:44:f3:17:f6:10:  
39:7f:e9:65:e8:31:3e:ca:dd:5f:d8:4e:1c:0a:42:76:ce:dc:



```
0b:12:7b:b9:14:f9:3d:ee:76:b5:34:ba:f7:60:f2:30:e3:d6:
55:dd:70:f0:9e:75:ff:0a:5c:4f:10:a7:ce:7b:a6:80:5d:8a:
18:bd:dd:18:58:95:f1:ae:ae:5d:2f:cc:5c:fe:a4:26:a2:7f:
5d:b8:51:7e:1f:3c:d6:d8:7d:65:02:7f:17:e2:d7:32:5d:e5:
99:7b:80:d0:2f:21:58:3e:74:ad:9b:35:dc:c9:f7:66:65:75:
36:8f:91:55:bb:33:68:41:cc:26:57:79:a3:e5:82:be:80:9b:
de:08:86:3d:74:2c:72:99:4c:b5:41:ed:5e:92:08:6b:56:2b:
58:56:e9:47:e7:c0:7c:c2:32:dc:04:90:37:bc:d1:d2:e5:8e:
0a:a1:a4:28:88:d5:b3:94:51:34:20:75:17:e6:d3:c8:9d:00:
f6:8c:8c:46:9b:53:30:ce:81:53:b6:52:72:26:c6:4d:76:50:
fc:0c:31:bf:09:9e:ee:ea:a4:8d:8f:b9:84:a4:45:b6:06:31:
25:06:c2:2b:6f:97:0a:84:7b:cb:bd:aa:45:7b:8e:04:96:5f:
d9:9a:30:86:9c:32:4b:89:4a:6c:e8:87:c8:d2:f6:6b:35:d5:
a1:e2:97:c6:3b:3a:02:54
```

-----BEGIN X509 CRL-----

```
MIIECzCCAfMCAQEWdQYJKoZIhvcNAQELBQAwUjEaMBGGA1UEAwwRUkNEZXZzIFN1
cHBvcnQgQ0ExCzAJBgNVBAsMAkiUMRowGAYDVQQKDBFSQ0RldnMgU3VwcG9ydCBT
QTElMAkGA1UEBhMCTFUXDTIzMTIxNDE1NDUyOVVwXDTI0MDEzMzE1NDUyOVVwgcgw
MAIRALo/XdZbhk76mLD0SE6YR4XDTIzMTIxNDE1NDUyOFowDDAKBgNVHRUEAwoB
BTAAwAhEA1OqSIU3vqTdrn+QVh0BYbxcNMjMxMjE0MTU0NTI4WjAMMAoGA1UdFQQD
CgEFMDACEQCwnoqE4ZYU1AsrSSNb4NQeFw0yMzEyMTQxNTQ1MjhaMAwwCgYDVR0V
BAMKAQUwMAIRANEFQmwmBBgYU86MrgFqPRkXDTIzMTIxNDE1NDUyOFowDDAKBgNV
HRUEAwoBBaCB0TCBnjCBjwYDVR0jBIGHMIGEgBS6xt28Ms5X3s4/ye1OjQhmv6nw
jKFWpFQwUjEaMBGGA1UEAwwRUkNEZXZzIFN1cHBvcnQgQ0ExCzAJBgNVBAsMAkiU
MRowGAYDVQQKDBFSQ0RldnMgU3VwcG9ydCBTQTElMAkGA1UEBhMCTFVCFDjJtCDY
JXiTIVqxh62ME0Foa0DMAoGA1UdFAQDAgEBMA0GCSqGSIb3DQEBCwUAA4ICAQBx
KKUNoFFzJmlqBY7ME7ZDfNxGC4ElzBY5SpavB9itRdta0TwuZVMH/x9F0ZzooOOc
Ejg7zRyRkPHRYHITOUrXsDcrXLihGy4qbRtDyZ57IYFwEa57dpN3b22tHgefx5t
XR8Vlt275xOWwZkBbqHRWkcnwasRt+sUJOp3yYHqzISGINh/9aUOV/sh7u3iU5cs
RwmsWRCKJRwpv2CpTT7kj6p9rYfWn3MwzlRbj7cJWA48t+7KbLzKD7mjNzXh+C0
IC0uhww8jqnBIQNw7hNXjjOIEzG3TkNxDTym3p8xcl/jiPVZ1v8hR0wuH2T4tKjY
Akl0JFTYRPMX9hA5f+ll6DE+yt1f2E4cCkj2ztwLEnu5FPk97na1NLR3YPlw49ZV
3XDwnnX/ClxPEKfOe6aAXYoYvd0YWJXxrq5dL8xc/qQmon9duFF+HzzW2H1lAn8X
4tcyXeWZe4DQLyFYPnStmzXcyfdmZXU2j5FVuzNoQcwmV3mj5YK+gjveCIY9dCxy
mUy1Qe1ekghrVitYVulH58B8wjLcBJA3vNHS5Y4KoaQoiNWzIFE0IHUX5tPlnQD2
jlxGm1MwzoFTtJjySZNdID8DDG/CZ7u6qSNj7mEpEW2BjEIBsIrb5cKhHvLvapF
e44Eil/ZmjCGnDjLiUps6lfl0vZrNdWh4pfGOzoCVA==
```

-----END X509 CRL-----

The following command can be utilized to convert the CRL file from DER format to PEM format:

```
openssl crl -inform DER -in webadm1.crl.der -outform PEM -out webadm1.crl
```

The output of the previous command is displayed below:

Certificate Revocation List (CRL):

Version 2 (0x1)

Signature Algorithm: sha256WithRSAEncryption  
Issuer: CN=RCDevs Support CA, OU=IT, O=RCDevs Support SA, C=LU  
Last Update: Dec 14 15:45:29 2023 GMT  
Next Update: Jan 13 15:45:29 2024 GMT  
CRL extensions:

X509v3 Authority Key Identifier:

keyid:BA:C6:DD:BC:32:CE:57:DE:CE:3F:C9:ED:4E:8D:08:67:BF:A9:F0:8C  
DirName:/CN=RCDevs Support CA/OU=IT/O=RCDevs Support SA/C=LU  
serial:32:49:B4:20:D8:25:78:93:95:5A:B1:87:AD:8C:13:43:85:A1:AD:03

X509v3 CRL Number:

1

Revoked Certificates:

Serial Number: BA3F5DD65B864EFA98B0F4484E98471E

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: D4EA92954DEFA9376B9FE4158740586F

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: B09E8A84E19614D40B2B49235BE0D41E

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Serial Number: D105426C2604181853CE8CAE016A3D19

Revocation Date: Dec 14 15:45:28 2023 GMT

CRL entry extensions:

X509v3 CRL Reason Code:

Cessation Of Operation

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

71:28:a5:0d:a0:51:73:26:62:2a:05:8e:cc:13:b6:43:7c:dc:  
46:0b:81:08:cc:16:39:4a:96:af:07:d8:ad:45:db:5a:d1:3c:  
2e:65:53:07:ff:1f:45:d1:9c:e8:a0:e3:9c:10:98:3b:cd:1c:  
91:90:f1:d1:60:79:53:39:4a:d7:49:d0:ab:5c:b9:61:1b:2e:  
2a:6d:1b:43:c9:9e:7b:95:86:05:c0:46:b9:ed:da:4d:dd:bd:  
b6:b4:78:1e:7f:1e:6d:5d:1f:15:2e:dd:bb:e7:13:96:c1:99:  
01:6e:a1:d1:5a:48:e7:c1:ab:11:b7:eb:14:24:ea:77:c9:81:  
ea:cc:84:86:20:d8:7f:f5:a5:0e:57:fb:21:ee:ed:e2:53:97:  
2c:47:09:ac:59:10:8a:25:1c:29:bf:60:a9:4d:3e:e4:8f:aa:  
7d:ad:87:d6:9f:73:30:23:39:51:6e:3e:dc:25:60:38:f2:df:  
bb:29:b2:f3:28:3e:e6:24:dc:d7:87:e0:b4:94:2d:2e:87:0c:  
3c:8e:a9:c1:95:03:70:ee:13:57:8c:93:a5:13:31:b7:4e:43:  
71:0d:3c:a6:de:9f:31:70:8f:e3:88:f5:59:d6:ff:21:47:4c:  
2e:1f:64:f8:b4:a8:d8:02:49:74:24:54:d8:44:f3:17:f6:10:  
39:7f:e9:65:e8:31:3e:ca:dd:5f:d8:4e:1c:0a:42:76:ce:dc:  
0b:12:7b:b9:14:f0:2d:ca:76:b5:24:b3:f7:60:f2:20:c2:d6:

```
00:12:70:b9:14:19:5d:ee:70:b3:34:ba:17:00:12:30:e3:00:
55:dd:70:f0:9e:75:ff:0a:5c:4f:10:a7:ce:7b:a6:80:5d:8a:
18:bd:dd:18:58:95:f1:ae:ae:5d:2f:cc:5c:fe:a4:26:a2:7f:
5d:b8:51:7e:1f:3c:d6:d8:7d:65:02:7f:17:e2:d7:32:5d:e5:
99:7b:80:d0:2f:21:58:3e:74:ad:9b:35:dc:c9:f7:66:65:75:
36:8f:91:55:bb:33:68:41:cc:26:57:79:a3:e5:82:be:80:9b:
de:08:86:3d:74:2c:72:99:4c:b5:41:ed:5e:92:08:6b:56:2b:
58:56:e9:47:e7:c0:7c:c2:32:dc:04:90:37:bc:d1:d2:e5:8e:
0a:a1:a4:28:88:d5:b3:94:51:34:20:75:17:e6:d3:c8:9d:00:
f6:8c:8c:46:9b:53:30:ce:81:53:b6:52:72:26:c6:4d:76:50:
fc:0c:31:bf:09:9e:ee:ea:a4:8d:8f:b9:84:a4:45:b6:06:31:
25:06:c2:2b:6f:97:0a:84:7b:cb:bd:aa:45:7b:8e:04:96:5f:
d9:9a:30:86:9c:32:4b:89:4a:6c:e8:87:c8:d2:f6:6b:35:d5:
a1:e2:97:c6:3b:3a:02:54
```

You can locate the CRL requests logs in the `/opt/webadm/logs/webadm.log` file after they have been downloaded.

```
[2023-12-13 17:30:09] [192.168.3.205:60062] New CRL request
[2023-12-13 17:30:09] [192.168.3.205:60062] Found 4 revoked certificates (cached)
```

## 17. API Keys

API keys has been introduced from WebADM 2.3 and is supported with all RCDevs plugins. Instead of using an SSL certificate for client authentication, you have the option to utilize an API key, which can serve as an alternative for secure communication between a client integration and a targeted web service (e.g. OpenOTP). It can be used with software deployed on-premise or in the cloud. One advantage of using an API key is that it potentially does not have an expiration date if you choose not to set one when issuing it but API Keys are considered as less secure than client certificate.

There is 2 possible ways to issue an API key for your client integration:

- > Through the WebADM Admin Portal;
- > Through the WebADM Manager API (upcoming versions);

### 17.1 Issue an API key

#### 17.1.1 Through WebADM Admin GUI

To create an API key in WebADM Admin GUI, follow these steps:

1. Log in to your WebADM Admin GUI as a `super_admin`.
2. Click on the “Admin” tab.
3. Select “Create Web Service API Key”.

[Home](#) | [Admin](#) | [Cluster](#) | [Create](#) | [Search](#) | [Import](#) | [Databases](#) | [Statistics](#) | [Applications](#) | [About](#) | [Logout](#)

### WebADM Server Administration

WebADM v2.3.0 (64bit) running on server webadm2.openotp (nodeid: 13ad823e) in **Cluster Mode**.

**Server Version Details:** WebADM/2.3.0 Apache/2.4.57 PHP/8.1.19 OpenSSL/1.1.1t  
**Internal Server Time:** 2023-05-19 11:31:09 Europe/Berlin (**NTP check Ok**)  
**Hardware Modules:** No HSM Configured  
**WebADM Features:** WebApps (**Enabled**), WebSrvs (**Enabled**), Manager (**Enabled**)  
**RCDevs Cloud Services:** BASE, PUSH, SMS, PROOF (**Connected**)

**Active LDAP Server:** LDAP Server 2 (10.10.2.4)      **Active SQL Server:** SQL Server 2 (10.10.2.4)  
**Active Session Server:** Session Server 1 (10.10.2.3)      **Active PKI Server:** PKI Server 2 (10.10.2.4)  
**Active Mail Server:** SMTP Server 2 (10.10.2.2)

**User Domains (1)**

Associate domain names with LDAP user search bases.

**Client Policies (1)**

Define custom policy settings for consumer applications.

**Hosted Tenants (7)**

Hosted customer tenant with dedicated LDAP subtree.

**LDAP Option Sets (0)**

LDAP subtree customizations, alerts and badging features.

**Administrator Roles (0)**

Create admin role templates for your 'other' administrators.

**LDAP Mount Points (0)**

Connect secondary LDAP servers to the tree view.

**Licensing and Configurations**

- [Software License Details](#)
- [LDAP Server Details](#)
- [LDAP Server Schema](#)
- [Memory Usage Details](#)
- [Hardware Modules Details](#)
- [Remote Manager Interface](#)
- [Config Object Statuses](#)
- [Network Service Statuses](#)
- [WebADM Base Settings](#)
- [Trusted CA Certificates](#)

**Runtime Actions**

- [Download Internal CA Certificate](#)
- [Download External CA Certificate](#)
- [Create Server or Client X.509 Certificate](#)
- [Create Web Service API Keys](#)
- [Clear System & Application Caches \(367 KB\)](#)
- [Clear Application Sessions & Work Data](#)
- [Start Scheduled Background Tasks](#)
- [Reload WebADM Configurations](#)
- [Send Test Alert Email](#)

You will be redirected to a new page where you need to provide the following information:

### Create Web Service API Key

You can use this form to issue API keys for Web services configured with the 'Require Certificate / API Key' option enabled. API keys can optionally be restricted to a specific application and can optionally auto-expire after a certain time.

API Key Description:

Restricted Application: [Not Set] v i

API Key expiration (in days): [Not Set] v

- > **API Key description** : Give a description that will help you identify the purpose of this API key.
- > **Restricted Application**(optional): You can restrict the usage of the API key to a specific Web Service.
- > **API Key Expiration** (optional): Set an expiration date for the API key. Once expired, the client application will be unable to communicate with the Web Service associated with this key. For example, with OpenOTP, an expired API key will prevent any further logins.

Once you have entered the required information, click the **Ok** button to generate the API key.

The API key will be generated, and you will see a confirmation message. Congratulations! Your API key has been successfully generated.

### 17.1.2 Through WebADM Manager API

Upcoming in next version.

## 17.2 Manage issued API keys

Once API keys have been issued, you have the ability to revoke them temporarily or permanently. You can also view the last usage of each API key, including its expiration time and the host IP of the last usage.

Database Viewer for Web Services API Keys (4 results out of 4 certificates)

Filters (0)

Reference  Equals  Add Filter

Not Expired Expired Enabled Revoked

Display Options: Retrieve max 1000, Page results 35, Refresh

Certificate Actions: Delete selected items, Delete expired items, Create new API key, Export as CSV / XML

<input type="checkbox"/>	Reference	Description	Application	Creation Time	Expiration Time	Last Use	Host IP	Access Token	Enabled
<input type="checkbox"/>	5711466305890178903	windows	OpenOTP	2023-05-19 11:56:51	2025-05-18 11:56:51	[NA]	[NA]	Copy PgCEcv07dFCzAe5N1ReL... Valid	<input checked="" type="checkbox"/>
<input type="checkbox"/>	4055845826231284548	CP12	OpenOTP	2023-05-17 17:01:40	[NA]	2023-05-17 17:04:31	213.135.242.3	Copy IsQsIAq2Ozsd6AIUlyRw... Valid	<input checked="" type="checkbox"/>
<input type="checkbox"/>	3552875366028103219	RDWEB	OpenOTP	2023-05-17 16:09:21	[NA]	2023-05-17 16:51:38	213.135.242.3	Copy FSvymYO7ZsqwWO5925qV... Valid	<input checked="" type="checkbox"/>
<input type="checkbox"/>	5280829322587550006	ADFS	OpenOTP	2023-05-17 14:20:52	[NA]	2023-05-17 15:15:02	213.135.242.3	Copy xuf7v1slXZKYQq3KMxGV... Valid	<input checked="" type="checkbox"/>

To temporarily revoke an API key, click the “Enabled” button associated with that key. This will disable the API key temporarily. If you wish to re-enable the key, simply click the same button again.

<input type="checkbox"/>	Reference	Description	Application	Creation Time	Expiration Time	Last Use	Host IP	Access Token	Enabled
<input type="checkbox"/>	5711466305890178903	windows	OpenOTP	2023-05-19 11:56:51	2025-05-18 11:56:51	[NA]	[NA]	Copy PgCEcv07dFCzAe5N1ReL... Valid	<input checked="" type="checkbox"/>
<input type="checkbox"/>	4055845826231284548	CP12	OpenOTP	2023-05-17 17:01:40	[NA]	2023-05-17 17:04:31	213.135.242.3	Copy IsQsIAq2Ozsd6AIUlyRw... Valid	<input checked="" type="checkbox"/>
<input type="checkbox"/>	3552875366028103219	RDWEB	OpenOTP	2023-05-17 16:09:21	[NA]	2023-05-17 16:51:38	213.135.242.3	Copy FSvymYO7ZsqwWO5925qV... Suspended	<input type="checkbox"/>
<input type="checkbox"/>	5280829322587550006	ADFS	OpenOTP	2023-05-17 14:20:52	[NA]	2023-05-17 15:15:02	213.135.242.3	Copy xuf7v1slXZKYQq3KMxGV... Valid	<input checked="" type="checkbox"/>

Delete an API key will revoke it permanently.

## 18. Managing Applications

WebADM registered applications provide their own configuration schemas to the system. The application configurations are accessible from the WebADM Applications menu. WebADM provides a very high-level interface for managing very complex application configurations. WebADM relies on XML schema files, which transparently make the mapping between the application configuration requirements and the graphical configuration editors. The schema files are provided with the applications and should not be edited.

To set up an application in WebADM:

1. Install the application on the system with its self-installer.
2. Enter WebADM and navigate to the Applications menu.
3. Click to register the application.

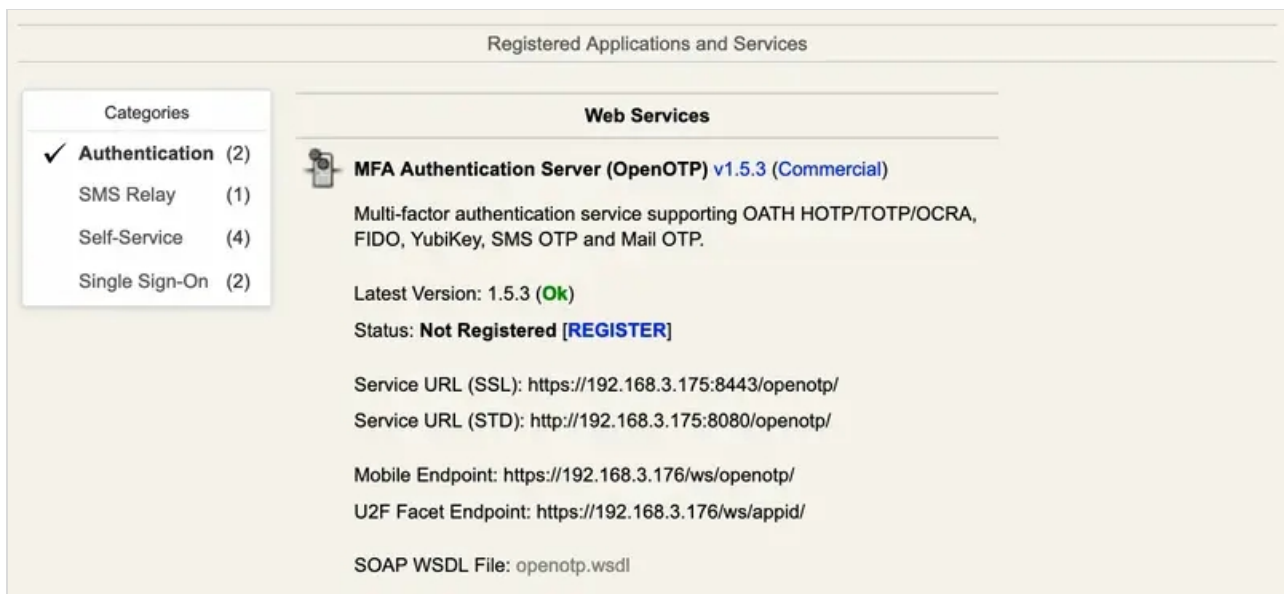


Figure 71. Register Application

4. Click to configure the application.

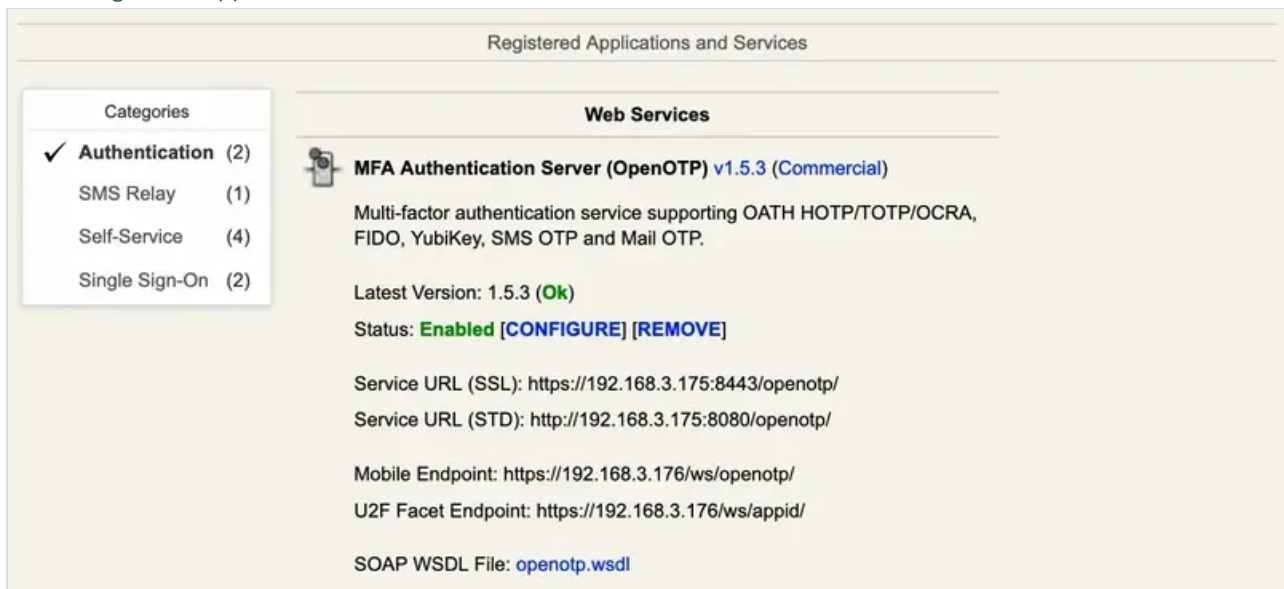


Figure 72. Configure Application

5. Save the settings and your application is now immediately operational.

## 18.1 User Application Settings

The default application's settings are defined as described just before. Yet, some settings can be re-defined per user or groups. WebADM processes the settings in the following order:

1. Application level settings are applied first. They are considered as default settings.
2. Group settings (if any) are applied. If the user is a member of multiple groups, the group's settings are merged.

**Note** User groups and group settings are cached for 5 minutes in order to optimize group searches and user setting resolutions. This has the side effect that user groups and group settings' changes may be delayed for a maximum time of 5 minutes when used by WebApps and Web Services.

3. User settings (if any) are applied.
4. Web Service Client settings are applied if the client requesting the service matches a Client object name.
5. Request settings (if any) are applied. The Web Service's API provides a Settings SOAP field to dynamically pass user settings to the WebADM services.

That means that the user settings have priority over the group settings which have themselves priority over the application default settings, etc...

To add settings to a user or group (when no setting is defined yet), select the WebADM Settings in the Add Attribute action for the user. If the user/group is not extended with the *webadmAccount* object class, or the group is not extended with the *webadmGroup* object class, you must extend it first with the Add Extension action to be able to add settings.

To modify the user/group settings in a *webadmAccount/webadmGroup* object, edit the object and click the links in the information box (at the top middle of the editorial page), or click the Edit WebADM Settings button in the object attribute list.

**WebADM Freeware Edition v2.0.7**  
Copyright © 2010-2020 RCDevs Security. All Rights Reserved

Home Admin Create Search Import Databases Statistics Applications About Logout

Application Settings for cn=marcus.o=Root

**Applications**

- ✓ **MFA Authentication Server**
- SSH Public Key Server
- Administration Help Desk
- OpenID & SAML Provider
- Secure Password Reset
- User Self-Service Desk
- User Self-Registration

**Authentication Policy**

**Login Mode** LDAPOTP (Default)

The login mode (required login factors) should be adjusted via Client Policies.

- LDAPOTP: Require both LDAP and OTP passwords.
- LDAPU2F: Require both LDAP and FIDO response.
- LDAPMFA: Require LDAP and either OTP or FIDO.
- LDAP: Require LDAP password only.
- OTP: Require OTP password only.

**OTP Type** TOKEN (Default)

- TOKEN: OATH HOTP/TOTP/OCRA, YubiKey or MobileOTP Token.
- SMS: SMS one-time password (On-demand or Prefetched).
- MAIL: Email one-time password (On-demand or Prefetched).
- LIST: Pre-generated OATH OTP password list (to be printed).
- VOICE: Voice biometrics authentication (requires license option).
- PROXY: Forward requests to another RADIUS server (for migrations).

**OTP Fallback** TOKEN

Fallback OTP Type to be used as secondary authentication method.  
SMS/MAIL OTPs are delayed for MobileTimeout seconds before being sent.  
LASTOTP let users use the last validated OTP which expires after a delay.  
Use DISABLED to disabled fallback if there is a configuration by default.

**OTP Password Length** 6 (Default)

Note: This setting is ignored for OCRA Tokens as OTP length is part of the OCRA Suite.  
Warning: Changing this setting after having registered OATH Tokens will invalidate these Tokens.

**OTP PIN Prefix**  Yes  No (default)

When enabled a static prefix has to be prepended to any OTP password in the form [PIN][OTP].  
The OTP Prefix must be registered first and must be at least 4 alpha-numeric characters.

**Mobile Response Timeout** 30 (Default)

Time to wait for mobile response with Token Simple Push before switching to a fallback method.  
Changing the default value requires to adjust client timeouts (ex. RADIUS request timeout) accordingly!  
Note: This timeout applies to MSS mobile responses as well (MobileID SMS delivery mode).

**Challenge Session Timeout** 90 (Default)

Timeout to wait for a challenge response (in seconds).  
Note: SMS OTP and MAIL OTP may requires longer timeouts.

**User Notifications**



Send Expire Notification MAIL

Send a notification email/SMS to the user when his LDAP password or OTP Token expired.  
The email subject and sender address are defined in the MAIL OTP Settings.  
The SMS sender number is defined in the SMS OTP Settings.

Send Blocking Notification MAIL

Send a notification email/SMS to the user when his account gets blocked.  
The email subject and sender address are defined in the MAIL OTP Settings.  
The SMS sender and message type are defined in the SMS OTP Settings.

Send Self-Registration Links  Yes  No (default)

Automatically send a self-registration email/SMS to the user has no Token registered or Token expired.  
This feature applies to the expiration of OTP List and Application Passwords too.  
Note: Requires the SelfReg WebApp to be installed.

Send Password Reset Links  Yes  No (default)

Automatically send a password reset email/SMS to the user password expired or must be changed.  
Note: Requires the PwReset WebApp to be installed.

---

#### Feature Activation

Enable MFA Login  Yes (default)  No

Login Allowed From  Edit

Login Allowed Until  Edit

---

#### User Blocking

Max Login Tries 0

Maximum number of authentication attempts before the account is blocked.  
Set '0' to disable blocking.

Block Time 0

Time to block a user account after Max Failed Passwords is reached (in seconds).  
Set '0' to block permanently.

Max Idle Time 0

Number of days after which an account is permanently blocked when unused.  
Set '0' to disable blocking.

---

#### Token Features

Simple-Push Login  Yes  No (default)

Enable simplified mobile Push authentication with one-tap 'Approve/Deny'.  
Simple-Push does not work as fallback and requires Mobile Push activation with RCDevs Authenticator.

Mobile Voice Login  Yes  No (default)

Use the Mobile Token for Voice login (does not require an OTP challenge)

**Token Expiration Time**

Time after which a Software Token expires and must be re-enrolled (in days).  
Set '0' to disable the expiration on newly registered software Tokens.

---

**OATH Tokens**

**HOTP Look Ahead Window**

Maximum amount of out-of-sync OTPs that can be generated before a manual resync is required.  
Warning: High values affect performances and security.

**TOTP Time Offset Window**

Maximum time offset (in seconds) between the server and the Token generator.  
Warning: High values affect performances and security.

**TOTP Time Step**

Time period or step (in seconds) for TOTP calculations.  
You can change the default value if you use Tokens with a different Time Step.  
Warning: Changing this setting after having registered TOTP Tokens will invalidate these Tokens.

**OCRA Suite**

OCRA suite can be customized according to OATH OCRA Suite specification.  
When counter is used, OCRA will use the HOTP Look Ahead Window setting above.  
When timestamp is used, OCRA will use the TOTP Time Offset Window setting above.  
Warning: Changing this setting after having registered OCRA Tokens will invalidate these Tokens.

---

**SMS OTP**

**SMS Message Type**

Flash (class 0) SMS are not stored on the mobile phone.

**SMS Delivery Mode**

Ondemand: A new OTP is sent when the user starts an authentication process.  
Prefetch: The next OTP is sent after the user performed an authentication.  
MobileID: Use Mobile Signature Service (MSS) instead of SMS OTP (not usable as SMS fallback).

**Prefetched OTP Expiration**

Time after which prefetched OTPs cannot be used anymore and must be re-generated (in days).  
This setting applies to prefetched Mail OTPs too.

---

**MAIL OTP**

**Use Secure Email**  Yes  No (default)

Encrypt OTP email with the user certificate public key (S-MIME).

**Email Delivery Mode**

Ondemand: A new OTP is sent when the user starts an authentication process.  
Prefetch: The next OTP is sent after the user performed an authentication.

---

**LIST OTP**

**List Size**

Number of OTP to be pre-generated.

**List Algorithm**

**List Challenge Mode**

Choose HideID not to display the expected OTP number to the user in the OTP challenge.

---

**LASTOTP Fallback**

**Last OTP Validity**

After this period (in seconds), the last saved OTP automatically expires.

**Last OTP Per IP**  Yes  No (default)

The last OTP is reusable only from the same IP address.

---

**RADIUS Options**

**RADIUS Reply Attributes**

RADIUS attributes to be sent to the VPN server or RADIUS client.  
You can return LDAP attribute values by configuring attribute values in the form 'LDAP:mobile'.

---

Figure 73. User Settings Editor

The Manager interface provides access to some WebADM user management functions and operations exported by your registered applications. The Manager also allows external systems such as Web portals to remotely trigger user management operations and actions from the network.

The user management functions provide LDAP operations such as object creation, update, removal, WebADM settings and data management, etc... The method names for internal management functions are in the form of *Manager\_Method*.

The operations exported by the registered applications provide access to any features which are accessible from the application actions in the Admin Portal. The method names for application-exported functions are in the form of *Application.Manager\_Method*.

The interface communication protocol is based on the JSON-RPC v2.0 specification. You can find the JSON-RPC specification at <http://jsonrpc.org/spec.html>.

You can go to the Manager Interface page in the WebADM Admin menu to have a full listing of the supported Manager functions and parameters. You can then navigate between applications to get the Manager functions supported by a specific registered application.

The Manager API requires authentication and a WebADM administrator account must be provided to access the interface. The authentication mechanism which is enforced is always the same as the mechanism configured for the WebADM Admin Portal (i.e. The `auth_mode` setting in the `webadm.conf` file).

#### Note

Any LDAP permission or OptionSet restriction configured in WebADM will be enforced within the Manager interface. Administrators have also the same level of access in the Manager as they have in the Admin Portal.

- › With DN login mode, the administrator DN and password must be provided in the HTTP-Basic Authorization header.
- › With UID login mode, the administrator user ID and password must be provided in the HTTP-Basic Authorization header.
- › With PKI login mode, the administrator's user certificate must be used for establishing the HTTPs connection to the interface and the administrator password must be provided in the HTTP-Basic Authorization header.

A connection to the Manager automatically creates an Administrator session in WebADM for processing the requested methods. The Manager responses return a session cookie called WEBADMMANAG in the response headers. You can pass the session cookie in the next Manager requests to avoid starting new sessions.

Note that the Manager sessions have a short expiration time and are automatically closed after 10 seconds of inactivity. Yet, you can force the closure of the session by passing the "Connection: close" header to the requests.

The Manager interface is accessible at the URL: <https://yourserver/manag/>.

Look at Appendix D for some simple examples of function calls using the PHP language to use the Manager Interface.

## 20. Installing WebApps and Web Services

WebADM has been designed to ease as much as possible the installations, upgrades and removal of applications (WebApps and Web Services).

To install a new RCDevs application (WebApp or Web Service) in WebADM, proceed as follows:

1. Get the application self-installer package from the RCDevs website.
2. Copy it to your Linux server running WebADM.
3. Uncompress it with the command `gunzip application-x.x.x.sh.gz`
4. Set installer as UNIX executable with the command `chmod 755 application-x.x.x.sh`.
5. Run the self-installer and answer the setup questions with the command `./application-x.x.x.sh`. WebApps application files will be installed in the `webapps/` system folder, under a folder having the name of your WebApp. Web Services application files will be installed in the `websrvs/` system folder, under a folder having the name of your Web Service.
6. Log in WebADM as with a super administrator account.
7. Navigate to the Applications menu and click the Register button for the new application (see section Applications Administrators for details). WebADM will create an LDAP configuration object for the new application in the `webapps_container` for WebApps and in the `websrvs_container` for Web Services, as defined in the WebADM main configuration file (`conf/webadm.conf`).



Figure 74. Application Registered in LDAP

8. Click the Configure button for the new application, adjust the application settings and save the settings (see section Applications Administrators for details). WebADM will update the LDAP configuration object with the new settings.

Important: You do not have to modify any file in the application installation directory! The applications configurations are managed and stored in LDAP by WebADM from the Applications menu only.

To upgrade an application, do not remove the previous version and proceed exactly like for the installation. Read the

CHANGELOG and README files to get the list of changes and proceed with the required modifications.

After a WebApp or Web Service upgrade, the application configurations may need to be updated. Log in WebADM and check the installed application status on the homepage or in the Applications menu. If a configuration update is required, click the *Not Configured* button to update the configuration and save the application settings again.

## 20.1 Embedding a WebApp

By default, WebApps are accessible from the WebApps portal at the URL `http://yourserver/webapps/`. And a specific WebApp (mywebapp) can be accessed at the URL `http://yourserver/webapps/mywebapp`.

You can embed a WebApp directly into a part of your website in an HTML iFrame or HTML Object. Insert the following code into your website to embed a WebApp directly into your website.

```
<object data="https://myserver/webapps/mywebapp?inline=1" />
```

Replace *myserver* and *mywebapp* with your WebADM server address and the WebApp name.

The parameter *inline=1* informs WebADM that the WebApp is embedded. WebADM will skip the HTML headers, footers and stylesheets. It will stream only the HTML BODY content for the WebApp.

## 21. Clustering

WebADM has been entirely designed for clustering. A WebADM system can be divided into more than one server for failover or load-balancing purposes. A WebADM server can be dedicated to one specific task such as administrator portal, WebApp server or Web Services servers. Moreover, multiple servers can be assigned the same task.

Please look at the WebADM High Availability Guide for Cluster installations.

For a clustered configuration, you mainly have to respect the following conditions:

1. All the servers of the cluster must use the same session manager at one moment. There can be multiple session managers for failover but all the systems must be configured to work with the same session manager at one time.
2. All the servers of the cluster must use the same PKI server (as for session manager). This is mandatory to keep the Certificate Authority consistent.
3. All the servers must have basically the same configurations and especially must use the same LDAP encryption key.

The session manager (Redis) and PKI server (RSignd) are very high-performance systems, multithreaded and written in C. Redis components do not call external network services such as LDAP or SQL servers and can also handle very high numbers of requests. Rsignd need LDAP and SQL server in order to store certificates.

Several servers can be configured to play the same role without any consequence because the application configurations and user information are stored in LDAP (which is a network service). So as long as all the clustered servers are connected to the same services (or multiple real-time replicated services) the whole system should not be impacted by the clustering.

Several Web Services servers can be accessed at the same time and client requests can even go randomly to any of the servers without a problem (as long as all the servers use the same session manager). For example, an OpenOTP SMSTOP login request can come to Server1, and the OTP challenge-response request can come to Server2. As Server1 and Server2 use the same session manager, the second request will be recognized by Server2 and part of a valid session.

In a clustered system, all the WebADM servers are automatically informed when an application configuration is changed by an administrator, and then, all the servers automatically refresh their configuration caches.

## Warning

Starting from WebADM version 1.4.2, any high availability and clustering feature require an RCDevs Enterprise license. Without a valid license file, the HA and cluster features are automatically disabled.

## 22. LDAP Permissions

### 22.1 WebADM Proxy User

There are two things to be considered in order to implement fine-grained LDAP permission for WebADM and its applications.

1. WebADM Proxy user permissions: This system user is used by WebADM to access and manipulate the required LDAP resources.
2. Administrator users permissions: These accounts login to the Admin portal in order to manage LDAP resources and registered applications.

The proxy user is required by WebADM to access LDAP resources (ex. application configuration, users, groups...) out the permissions of an Admin user's session.

The proxy user must have at least read-only permissions on the whole LDAP tree. It is used by the WebApps and Web Services such as OpenOTP and also requires some attribute write permissions as described below, over the trees where are stored the LDAP users.

By default, and for simplification, it is recommended to use an Administrator account of the LDAP directory as WebADM Proxy user.

If you need to implement finer LDAP access rights then:

1. The proxy user needs to perform a wide LDAP search and reads. It also requires read-only permissions to the WebADM LDAP configurations (i.e. configured containers) and to the user Domains subtrees.
2. The proxy user needs to do some write operations to a few LDAP attributes because it needs to store dynamic application user data into the users.

In some circumstances, the Proxy user will also need to write an application setting on the users and groups. The following attributes are part of the WebADM LDAP schema and need Proxy user write permissions:

- > `webadmData`: is the attribute where the applications store the user data (ex. OpenOTP enrolled Token states).

> `webadmSettings`: is the attribute where WebADM stores user-specific settings (ex. per-user OTP policy).

If you use WebADM Self-Services and depending on what you allow users to do within the self-service applications, then WebADM Proxy user may need some additional permissions: Ex. if you want users to reset their LDAP password, set their mobile numbers or email addresses, then the Proxy user will need to have write permissions to the corresponding LDAP attributes.

In general, it is recommended to implement Proxy user write access to the following attributes:

- > `webadmData` (dynamic and encrypted application data)
- > `webadmSettings` (only if Self-Services are used to configure account settings)
- > `mail` (only if Self-Services are used to set email addresses)
- > `mobile` (only if Self-Services are used to set mobile numbers)
- > `preferredLanguage` (only if Self-Services are used to set user language)
- > `userPassword` or `unicodePwd` for Windows AD (only if Self-Services are used to set user password)

## 22.2 Administrators

When an administrator logs in the WebADM Admin Portal, he always accesses and manages the LDAP resources under his own LDAP permissions. This means the user/group/configuration management permissions are enforced at the LDAP level. For example, a Windows AD Domain Administrator will be able to manage users and groups.

### Note

To be able to log in WebADM, an LDAP user must be either a Super Administrator (configured in `super_admins` in `webadm.conf`) or another Administrator (delegated administrator). Another Administrator is any admin users which is part of a WebADM Admin Role.

## 23. Using Custom SSL Certificates

The WebADM setup script automatically creates a self-signed CA certificate for the PKI server and a self-signed SSL certificate which is used by both the WebADM's HTTP and RSignd services. The SSL certificate file is stored in `/opt/webadm/pki/webadm.crt` and the corresponding key file is stored in `/opt/webadm/pki/webadm.key`.

Yet, in WebADM it is possible to use another (external) SSL certificate. This is useful if you need your WebADM HTTP services to operate under a trusted certificate. To use a custom SSL certificate, you need the certificate and key files from your CA vendor in PEM format. The certificate file may optionally contain the intermediate CA certificate list (concatenated after the PEM data). The custom certificate file must be stored in `/opt/webadm/pki/custom.crt` and the key file must be stored in `/opt/webadm/pki/custom.key`. The custom certificate will be used by the WebADM Admin Portal and the WebApps only. WebADM Web services will still operate with the self-signed SSL certificate. You must also not remove the `webadm.crt` and `webadm.key` files. These certificate files are still used by RSignd and your SOAP Web services.

Note: If you configure a CA certificate trust for your Web services' integrations (ex. OpenOTP integrations plugins), the trusted CA

certificate is always the WebADM's internal PKI certificate which you can download under the WebADM Admin menu.

## Appendix A: Sample webadm.conf File

```
#
# WebADM Server Configuration
#

# Administrator Portal's authentication method.
# - PKI: Requires client certificate and login password.
# - UID: Requires domain name, login name and password.
# - DN: Requires login DN and password.
# - OTP: Like UID with an OTP challenge.
# - U2F: Like UID with a FIDO-U2F challenge.
# - MFA: Like UID with both OTP and FIDO-U2F challenge.
# Using certificates is the most secure login method. To use certificate login,
# you must log in WebADM and create a login certificate for your administrators.
# The UID mode requires a WebADM domain to exist and have its User Search Base
# set to the subtree where are located the administrator users. When using UID
# and if there is no domain existing in WebADM, the login mode is automatically
# forced to DN. You will also need to log in with the full user DN and set up
# a WebADM domain to be able to use the UID login mode.
admin_auth UID

# Show the registered domain list when admin_auth is set to UID, OTP or U2F.
# And set a default admin login domain when auth_mode is set to these methods.
list_domains Yes
#default_domain "Default"

# Manager API's authentication method. Only UID, PKI and DN are supported here.
# If you set the admin_auth with multi-factor (PKI, OTP or U2F), then you must
# either use manager_auth PKI or UID with a list of allowed client IPs.
#manager_auth UID
#manager_clients "192.168.0.10","192.168.0.11"

# User level changes the level of feature and configuration for all applications.
# WebADM proposes three levels: Beginner, Intermediate and Expert. The default
# level (Expert) is recommended as it provides access to all the RCDevs features.
user_level Expert

# If your LDAP directory is setup with a base DN (ex. dc=mydomain,dc=com on AD),
# you can optionally set the base_treebase suffix and omit the suffix in other
# LDAP configurartions like proxy_user, super_admins and containers.
#ldap_treebase "dc=mydomain,dc=com"

# The proxy user is used by WebADM for accessing LDAP objects over which the
# admin user does not have read permissions or out of an admin session.
# The proxy user should have read permissions on the whole LDAP tree,
# and write permissions on the users/groups used by the WebApps and WebSrvs.
```



```

# The use of a proxy user is required for WebApps and WebSrvs.
# With ActiveDirectory, you can use any Domain Administrator DN as a proxy user,
# which should look like cn=Administrator,cn=Users,dc=mydomain,dc=com.
proxy_user    "cn=webadm,dc=WebADM"
proxy_password "Password1234"

# Super administrators have extended WebADM privileges such as setup permissions,
# additional operations and unlimited access to any LDAP encrypted data. Access
# restriction configured in the WebADM OptionSets and AdminRoles do not apply to
# super admins. You can set a list of individual LDAP users or LDAP groups here.
# With ActiveDirectory, your administrator account should be is something like
# cn=Administrator,cn=Users,dc=mydomain,dc=com. And you can replace the sample
# super_admins group on the second line with an existing security group.
super_admins "cn=admin,o=root", \
    "cn=super_admins,dc=WebADM"

# LDAP objectclasses
container_oclasses    "container", "organizationalUnit", "organization", "domain", "locality", \
    "country", "openldaprootdse", "treeroot"
# user_oclasses is used to build the LDAP search filter with 'Domain' auth_mode.
# If your super admin user does not have one of the following objectclasses,
# add one of its objectclasses to the list.
user_oclasses        "user", "account", "person", "inetOrgPerson", "posixAccount"
group_oclasses       "group", "groupOfNames", "groupOfUniqueNames", "groupOfURLs", "posixGroup"
# With ActiveDirectory 2003 only, you need to add the 'user' objectclass to the
# webadm_account_oclasses and the 'group' objectclass to the webadm_group_oclasses.
webadm_account_oclasses "webadmAccount"
webadm_group_oclasses  "webadmGroup"
webadm_config_oclasses "webadmConfig"

# LDAP attributes
certificate_attrs    "userCertificate"
password_attrs      "userPassword", "unicodePwd", "sambaNTPassword"
uid_attrs           "uid", "samAccountName", "userPrincipalName"
member_attrs       "member", "uniqueMember"
memberof_attrs     "memberOf", "groupMembership"
memberuid_attrs    "memberUid"
language_attrs     "preferredLanguage"
mobile_attrs       "mobile"
mail_attrs         "mail"
webadm_data_attrs  "webadmData"
webadm_settings_attrs "webadmSettings"
webadm_type_attrs  "webadmType"
webadm_voice_attrs "webadmVoice"

# Set the LDAP container required by WebADM to store its configuration objects.
config_container "dc=WebADM"

# You can alternatively configure each configuration container independently.
# "dc=webadm,dc=WebADM"

```

```
#domains_container "dc=Domains,dc=WebADM"
#clients_container "dc=Clients,dc=WebADM"
#devices_container "dc=Devices,dc=WebADM"
#webapps_container "dc=WebApps,dc=WebADM"
#websrvs_container "dc=WebSrvs,dc=WebADM"
#adminroles_container "dc=AdminRoles,dc=WebADM"
#optionsets_container "dc=OptionSets,dc=WebADM"
#mountpoints_container "dc=MountPoints,dc=WebADM"

# You can set here the timeout (in seconds) of a WebADM session.
# Web sessions will be closed after this period of inactivity.
# The Manager Interface cookie-based sessions are disabled by default.
admin_session 900
manager_session 0
webapps_session 600

# You can set here the WebADM internal cache timeout. A normal value is one hour.
cache_timeout 3600

# Application languages
languages "EN","FR","DE","ES","IT","FI"

# WebADM encrypts LDAP user data, sensitive configurations and user sessions with
# AES-256. The encryption key(s) must be 256bit base64-encoded random binary data.
# Use the command 'openssl rand -base64 32' to generate a new encryption key.
# Warning: If you change the encryption key, any encrypted data will become invalid!
# You can set several encryption keys for key rollout. All the defined keys are used
# for decrypting data. And the first defined key is used to (re-)encrypt data.
# Two encryption modes are supported:
# Standard: AES-256-CBC (default)
# Advanced: AES-256-CBC with per-object encryption (stronger)
encrypt_data Yes
encrypt_mode Standard
encrypt_hsm No
encrypt_key "cq19TEHgHLQuO09DXzjOw30rrQDLsPKT3NiL6I3BH2w="

# Hardware Cryptography Module
# Yubico YubiHSM and SCHSM are currently supported for hardware encryption.
# Up to 8 HSM modules can be concurrently attached to the server.
#hsm_model YubiHSM
#hsm_keyid 0
#hsm_pincode XXXXXX

# The data store defines which back-end is used for storing user data and settings.
# By default WebADM stores any user and group metadata in the LDAP objects. By setting
# the data_store to SQL, these metadata are stored in a dedicated SQL table.
# LDAP remains the preferred option because it maximizes the system consistency.
# SQL should be used only if you need read-only LDAP access for the proxy_user.
data_store LDAP
```

```
# The record store defines which back-end is used to store SpanKey records.
# Choose SQL to store records in the database and NAS to store on a shared NAS folder.
# With NAS, the store_path must be configured and accessible from all cluster nodes.
record_store SQL
#record_path "/mnt/records"

# The group mode defines how WebADM will handle LDAP groups.
# - Direct mode: WebADM finds user groups using the memberof_attrs defined above.
# In this case, the group membership is defined in the LDAP user objects.
# - Indirect mode: WebADM finds user groups by searching group objects which contain
# the user DN as part of the member_attrs.
# - Auto: Both direct and indirect groups are used.
# - Disabled: All LDAP group features are disabled in WebADM.
# By default (when group_mode is not specified) WebADM handles both group modes.
group_mode Auto

# LDAP cache increases a lot of performances under high server loads. The cache limits
# the number of LDAP requests by storing resolved user DN and group settings. When
# enabled, results are cached for 300 secs.
ldap_cache Yes

# LDAP routing enables LDAP request load-balancing when multiple LDAP servers are
# configured in servers.xml. You should enable this feature only if the LDAP server
# load becomes a bottleneck due to a big amount of users (ex. more than 10000 users).
#ldap_routing No

# You can optionally disable some features if you run multiple WebADM servers with
# different purposes. For example, if you don't want to provide admin portal on an
# Internet-exposed WebApps and WebSrvs server.
# By default, all the functionalities are enabled.
enable_admin Yes
enable_manager Yes
enable_webapps Yes
enable_websrvs Yes

# Enable syslog reporting (disabled by default). When enable, system logs are sent
# to both the WebADM log files and syslog.
#log_debug No
#log_mixsql No
#log_syslog No
#syslog_facility LOG_USER
#syslog_format CEF

# Alerts are always recorded to the SQL Alert log. Additionally, when alert_email
# or alert_mobile is defined, the alerts are also sent by email/SMS.
#alert_email "me@mydomain.com"
#alert_mobile "+33 12345678"

# Alert users via email when a login certificate or ActiveDirectory domain password
# is near expiration. The templates are defined in ldap_expire_www and cert_expire_www
```

```
# is near expiration. The templates are defined in ldap_expire_XXX and cert_expire_XXX.
user_warning Yes

# Protect WebADM against bruteforce attacks on the WebApps by blacklisting source IPs
# for 20 seconds after 5 failed login attempts.
ip_blacklist Yes

# You can publish WebADM applications and OpenOTP mobile endpoint over Internet using
# a reverse proxy (WAF) or RCDevs WebADM Publishing Server (WAProxy).
# Set the IP address(es) of your reverse-proxy or WAProxy server(s). WebADM expects
# the HTTP_X_FORWARDED_FOR and HTTP_X_FORWARDED_HOST headers from reverse proxies!
# Use 'waproxy_proxies' ONLY if you are using RCDevs WAProxy as reverse-proxy!
#reverse_proxies "192.168.0.100", "192.168.0.101"
#waproxy_proxies "192.168.0.102"
# The 'public_hostname' is mandatory to let WebADM know your public endpoints' URLs.
# Use the public DNS name of your reverse proxy or WAProxy server without a scheme.
# The setting used to be named 'waproxy_pubaddr' in WebADM versions before v2.3.12.
#public_hostname "www.myproxy.com"

# Check for new product versions and license updates on RCDevs' website.
# These features require outbound Internet access from the server.
cloud_services Yes

# WebApps theme (default or flat)
# Comment the following line to disable the default theme.
webapps_theme "default"

# End-user message templates
# The following variables are available: %USERNAME%, %USERDN%, %USERID%, %DOMAIN%,
%APPNAME%
# Additional variables are available depending on the context: %APPNAME%, %APPID%, %TIMEOUT%,
%EXPIRES%
app_unlock_subject "Unlocked access to %APPNAME%"
app_unlock_message "Hello %USERNAME%,\r\n\r\nYou have a one-time access to the
%APPNAME%.\r\nYour access will automatically expire %EXPIRES%."
ldap_expire_subject "Login password near expiration"
ldap_expire_message "Hello %USERNAME%,\r\n\r\nYour login password will expire %EXPIRES%.\r\nPlease
reset your password before expiration!\r\n\r\nRegards"
cert_expire_subject "Login certificate near expiration"
cert_expire_message "Hello %USERNAME%,\r\n\r\nYour login certificate will expire %EXPIRES%.\r\nPlease
renew your certificate before expiration!\r\n\r\nRegards"

# Personalization options
# You can customize your organization's name, logo file and website URL.
# The logo file must be a PNG image under conf/ with a size of 100x50 pixels.
#org_name "RCDevs SA"
#org_logo "rcdevs.png"
#org_site "http://www.rcdevs.com/"
#org_from "noreply@rcdevs.com"
```

```
# Misc options
#treeview_width 300
#treeview_items 1500
#default_portal Admin
#ldap_uidcase No
#ntp_server "myserver"
```

## Appendix B: Sample servers.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<Servers>
```

```
<!--
```

```
*****
```

```
*** WebADM Remote Server Connections ***
```

```
*****
```

You can configure multiple instances for each of the following servers. At login, WebADM will try to connect the configured servers in the same order they appear in this file and uses the first one it successfully establishes the connection to. If the server connection goes down, it will automatically failover to the next configured server.

Any special characters must be encoded in XML compliant format. At least one LDAP server and one SQL server is required to run WebADM. Supported servers: OpenLDAP, Active Directory, Novell eDirectory, 389.

Allowed LDAP parameters are:

- name: server friendly name
- host: server hostname or IP address
- port: LDAP port number
  - default and TLS: 389
  - default SSL: 636
- encryption: connection type
  - allowed type are NONE, SSL and TLS
  - default: 'NONE'
- ca\_file: Trusted CA for SSL and TLS
- cert\_file: client certificate file
- key\_file: client certificate key
- sasl: SASL bind options separated by spaces

```
-->
```

```
<LdapServer name="LDAP Server"
```

```
host="localhost"
```

```
port="389"
```

```
encryption="NONE"
```

```
ca_file=""
```

```
sasl="" />
```

```
<!--  
<LdapServer name="LDAP Server 2"  
  host="remotehost"  
    port="389"  
  encryption="TLS"  
  ca_file="" />  
-->
```

```
<!--  
SQL servers are used for logs; message localizations and inventories.  
Supported servers: MySQL5, MySQL8, PostgreSQL, MSSQL, Sybase, Oracle, SQLite.
```

Allowed SQL parameters are:

- type: MySQL5, MySQL8, MariaDB, PostgreSQL, MSSQL, Sybase, Oracle or SQLite.
- name: server friendly name
- host: server hostname or IP address
- port: SQL port number (depends on server type)
- user: database user
- password: database password
- database: database name
- encryption: connection type allowed type are NONE, SSL and TLS
- ca\_file Trusted CA for SSL and TLS
- cert\_file: client certificate file
- key\_file: client certificate key

With SQLite, only the 'database' must be set and other parameters are ignored. The database is the full path to an SQLite DB file where WebADM has full write access.

With Oracle, you can optionally use TNS names. If the 'tnsname' is set then the 'host' and 'port' parameters are ignored and a tnsnames.ora file must exist under the conf/ directory.

```
-->
```

```
<SqlServer name="SQL Server"  
  type="MySQL"  
  host="localhost"  
  user="webadm"  
  password="webadm"  
  database="webadm"  
  encryption="NONE" />
```

```
<!--  
A session server is required for web services using sessions  
such as OpenOTP. You can specify one or more SQL servers here.  
The session server is included in WebADM. So you can keep the  
default settings here.
```

```
Using SSL/TLS encryption requires port 4001 and may slow down
```

Using SSL/TLS encryption requires port 4001 and may slow down your WebADM execution by a factor of 20%.

WARNING: TLS support is currently broken with PHPRedis!

-->

```
<SessionServer name="Session Server"
  host="localhost"
  port="4000"
  encryption="NONE"
    secret="secret" />
```

<!--

A PKI server (or CA) is required for signing user certificates. The RSign PKI server is included in WebADM. So you can keep the default settings here.

-->

```
<PkiServer name="PKI Server"
  host="localhost"
  port="5000"
  secret="secret" />
```

<!--

HTTP proxy servers can be used by WebADM for connecting remote Web services and version checking.

-->

<!--

```
<ProxyServer name="HTTP Proxy"
  host="proxy"
    port="8080"
    user=""
    password=""
  ca_file="" />
```

-->

<!--

SMTP mail servers can be used by WebADM for sending emails. If no server is specified, WebADM will use the local mailer in /usr/sbin/sendmail to send emails.

-->

<!--

```
<MailServer name="SMTP Server"
  host="localhost"
  port="25"
  user=""
  password=""
  encryption="NONE"
  ca_file="" />
```

-->

</Servers>

## Appendix C: Sample rsignd.conf File

```
#
# WebADM PKI Server Configuration
#

# Log file
logfile /opt/webadm/logs/rsignd.log
pidfile /opt/webadm/temp/rsignd.pid

# Default validity period for new certificates (in days)
# The CSR signing requests may set the validity period.
user_cert_validity 365
client_cert_validity 1825
server_cert_validity 3650

# Certificate and key used for the SSL listener
rsignd_cert /opt/webadm/pki/webadm.crt
rsignd_key /opt/webadm/pki/webadm.key

# Path CA certificate files and serial
ca_cert /opt/webadm/pki/ca/ca.crt
ca_key /opt/webadm/pki/ca/ca.key
ca_serial /opt/webadm/pki/ca/serial

# Serial number format (hex or dec)
serial_format hex

# Set to yes if the CA or RSignd private keys requires a decryption password.
# PEM passwords will be prompted at WebADM startup.
ca_password no
rsignd_password no

# HSM certificate authority (CA)
# The HSM model and PIN code are configured in webadm.conf.
hsm_ca no
hsm_keyid 0

#
# Directory or file containing trusted CA certificates (in PEM format)
# After adding a new certificate, type a "make" in the "trusted_ca_path"
# to rebuild certificate's hash.
# This is needed for rsignd to read the trusted CA certificates.
# Comment "trusted_path" to disable rsignd certificate's trust restrictions.
```



```
trusted_path /opt/webadm/pki/trusted
```

```
#  
# Client sections  
#  
# Declare here the Rsign clients with IP addresses or hostnames.  
# In cluster mode, the client WebADM server(s) must be defined here!
```

```
client {  
    hostname localhost  
    secret secret  
}
```

```
#client {  
#    hostname remote_server  
#    secret secret  
#}
```

## Appendix D: Sample Manager Interface Usage

Find below a few simple examples of the use of the WebADM Manager interface. The examples are written in PHP and use the cURL extension to send the JSON-RPC call over HTTP.

1. Resolve the DN of an existing user.

```

<?php
$method = 'Get_User_DN';
$params = array(
    'username' => 'test',
    'domain' => 'Default',
);

$request = array(
    'jsonrpc' => "2.0",
    'method' => $method,
    'params' => $params,
    'id' => 0);
$json = json_encode($request);

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "https://localhost/manag/");
curl_setopt($ch, CURLOPT_USERPWD, "default\\admin:password");
curl_setopt($ch, CURLOPT_HTTPHEADER, array("connection: close"));
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
$out = curl_exec($ch);
curl_close($ch);

print_r(json_decode($out));
?>

```

The manager will return a structure int form:

```

stdClass Object
(
    [jsonrpc] => 2.0
    [result] => cn=test,o=Root
    [id] => 0
)

```

2. Search email for LDAP users with the webadmAccount extension.

```

$method = 'Search_LDAP_Objects';
$params = array(
    'basedn' => 'o=root',
    'filter' => '(objectclass=webadmaccount)',
    'attrs' => array('mail')
);

```

Will return:

```
stdClass Object
(
    [jsonrpc] => 2.0
    [result] => stdClass Object
        (
            [cn=test1,o=Root] => stdClass Object
                (
                    [mail] => stdClass Object
                        (
                            [0] => test1@mycompany.com
                        )
                    )
            [cn=test2,o=Root] => stdClass Object
                (
                    [mail] => stdClass Object
                        (
                            [0] => test2@mycompany.com
                        )
                    )
            [id] => 0
        )
)
```

3. Set the user mobile number and email address.

```
$method = 'Set_User_attrs';
$params = array(
    'dn' => 'cn=test,o=root',
    'attrs' => array('mobile' => array('12345678'), 'mail' => array('test@test.com')),
);
```

Will return:

```
stdClass Object
(
    [jsonrpc] => 2.0
    [result] => 1
    [id] => 0
)
```

4. Get the user mobile number and email address.

```
$method = 'Get_User_attrs';
$params = array(
  'dn' => 'cn=test,o=root',
  'attrs' => array('mobile', 'mail'),
);
```

Will return:

```
stdClass Object
(
  [jsonrpc] => 2.0
  [result] => stdClass Object
  (
    [mobile] => Array
    (
      [0] => 12345678
    )

    [mail] => Array
    (
      [0] => test@test.com
    )

  )

  [id] => 0
)
```

5. Set some user application settings.

```
$method = 'Set_User_Settings';
$params = array(
  'dn' => 'cn=test,o=root',
  'settings' => array('OpenOTP.LoginMode' => 'LDAPOTP', 'OpenOTP.SecureMail' => false),
);
```

Will return:

```
stdClass Object
(
  [jsonrpc] => 2.0
  [result] => 1
  [id] => 0
)
```

6. Register a HOTP Token with OpenOTP.

```
$method = 'OpenOTP.HOTP_Register';
$params = array(
  'dn' => 'cn=test,o=root',
  'key' => base64_encode("12345678901234567890"),
  'counter' => 0
);
```

Will return:

```
stdClass Object
(
  [jsonrpc] => 2.0
  [result] => 1
  [id] => 0
)
```

#### 7. Create a WebADM-enabled user.

```
$method = 'Create_LDAP_Object';
$params = array(
  'dn' => 'cn=test_user,o=root',
  'attrs' => array('objectclass' => array('person','inetorgperson','webadmaccount'),
    'uid' => array('test_user'),
    'userpassword' => array('password'),
    'sn' => array('Test User'))
);
```

Will return:

```
stdClass Object
(
  [jsonrpc] => 2.0
  [result] => 1
  [id] => 0
)
```

#### 8. Create an Administrator user and add home to the admin group. In this example, we send two RPC commands in one single request.

```

$method = 'Create_LDAP_Object';
$params = array(
    'dn' => 'cn=test_admin,o=root',
    'attrs' => array('objectclass' => array('person','inetorgperson'),
        'uid' => array('test_admin'),
        'userpassword' => array('password'),
        'sn' => array('Test Admin'))
);
$request1 = array(
    'jsonrpc' => "2.0",
    'method' => $method,
    'params' => $params,
    'id' => 1
);

$method = 'Set_User_Attrs';
$params = array(
    'dn' => 'cn=other_admins,dc=WebADM',
    'attrs' => array('member' => array('cn=test_admin,o=root')),
    'values' => true
);
$request2 = array(
    'jsonrpc' => "2.0",
    'method' => $method,
    'params' => $params,
    'id' => 2
);

$request = array($request1, $request2);

```

Will return:

```

Array
(
    [0] => stdClass Object
        (
            [jsonrpc] => 2.0
            [result] => 1
            [id] => 1
        )
    [1] => stdClass Object
        (
            [jsonrpc] => 2.0
            [result] => 1
            [id] => 2
        )
)

```

## 9. Change a user password.

```
$method = 'Set_User_Password';  
$params = array(  
    'dn' => 'cn=test,o=root',  
    'password' => 'newpassword'  
);
```

Will return:

```
stdClass Object  
(  
    [jsonrpc] => 2.0  
    [result] => 1  
    [id] => 0  
)
```

*This manual was prepared with great care. However, RCDevs Security S.A. and the author cannot assume any legal or other liability for possible errors and their consequences. No responsibility is taken for the details contained in this manual. Subject to alternation without notice. RCDevs Security S.A. does not enter into any responsibility in this respect. The hardware and software described in this manual is provided on the basis of a license agreement. This manual is protected by copyright law. RCDevs Security S.A. reserves all rights, especially for translation into foreign languages. No part of this manual may be reproduced in any way (photocopies, microfilm or other methods) or transformed into machine-readable language without the prior written permission of RCDevs Security S.A. The latter especially applies for data processing systems. RCDevs Security S.A. also reserves all communication rights (lectures, radio and television). The hardware and software names mentioned in this manual are most often the registered trademarks of the respective manufacturers and as such are subject to the statutory regulations. Product and brand names are the property of RCDevs Security. © 2024 RCDevs Security S.A., All Rights Reserved*