WEBADM ADMINISTRATOR GUIDE V1

WebADM Administrator Guide v1

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 simplify 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-explaining.

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 ActiveDirectory 2008 & 2012, 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 ActiveDirectory, 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 ActiveDirectory domains and UNIX PAM-LDAP users. You can seamlessly manage both systems from the interface. Better, WebADM can extend your ActiveDirectory 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 object class 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 login certificates.

⚠ 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.

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 (ie. 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 and Redis database dump files.

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

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 >= 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 display 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.

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 the 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.

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

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 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.

- `other_admins`: This configuration has been removed in WebADM 1.4! Other administrators and admin users who are part of a WebADM Admin Role (See Admin Roles section for details). With WebADM 1.3 any non-super admin WebADM user must be defined in the other_admins to be able to log in and you can set a list of individual LDAP users or LDAP groups. You can comment on the setting not to use other administrators.
- `container_oclasses`: List of LDAP object classes to be considered by WebADM as LDAP containers.
- `user_oclasses`: List of LDAP object classes to be considered by WebADM as LDAP users. user_oclasses 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_oclasses`: List of LDAP object classes to be considered by WebADM as LDAP groups.
- `webadm_account_oclasses`: 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_oclasses`: 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_oclasses`: 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`: You can set here the timeout (in seconds) of a WebADM session. Sessions will be closed after this period of inactivity.
- `cache_timeout`: You can set here 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 (ie. 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 enable, 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.
- `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.
- `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.
- `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 rsignd.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}ZuWw1Ie2qxlguTF77mDjmQ==. You can use the new password value as is (ex. proxy_password "{wcrypt}ZuWw1Ie2qxlguTF77mDjmQ==").

> **Note**
>
> This feature requires an Enterprise License and 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 Enterprise 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.

Figure 12. Group Membership User Attribute

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

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.

Figure 14. Static Group

2. 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.

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.
- 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 on himself. 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 (Rsign) is the default certificate management system. However, it is also possible to use an external Certificate Authority (CA). In that case, WebADM will issue certificate signing requests (CSR) and ask for external signing.

Note

Any bindable object is able to log into WebADM. 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 *Domain, Trust, Client, AdminRoles, OptionSet, MountPoint, WebApp or WebSrv*.

Figure 19. Configuration Objects

WebADM creates a set of LDAP containers and objects during its setup for storing WebADM *Domains, OptionSets, MountPoints*, 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.

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. 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 Trust Domains

Trusts are special domain objects which do not correspond to a set of local LDAP users but to a remote WebADM installation's domain. The trust system works like a web service proxy for remote domains (within a trusted organization) and maps a local virtual domain name to a remote domain on another WebADM server.

Trusts work with all web services (OpenOTP, OpenSSO, SMSHub…).

6.2.6 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.

6.2.7 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 WebApp and Web Service 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.

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 super administrators and the users themselves are able to read this attribute unencrypted. They can edit the data values with the data editor.

> **Important**
>
> 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 LDAF 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.

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_oclasses*, 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_oclasses*, 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.

Figure 25. mobile Attribute

6.3.8 Mail Attribute

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

Figure 26. mail Attribute

6.3.9 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*).

Figure 27. webadmType Attribute

6.4 WebADM LDAP Schema

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

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 setup 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 the 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 auto-filled if default values are defined in the *OptionSets* which apply on the object creation context.

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 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).

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.

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`).

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.

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.

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.

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.

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.

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.

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.

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.

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 it's 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.

- `Certificate Signing Method`: Determines how WebADM will issue X.509 certificates. The available signing methods are:
  - Rsign, where WebADM uses its internal PKI subsystem to sign certificates (Default).
  - External, where WebADM uses HTML forms for copy-pasting data from an external Certification Authority (CA).
- `WebADM Account Quota`: Defines the maximum number of activated WebADM accounts within a context that can be created by the administrators. When quotas are defined at several tree levels, WebADM enforces the quotas from the current context down to the root level. In other words, if you have defined a quota of 10 activated accounts for context o=MyCompany, then its sub-contexts (such as ou=Sales, ou=Marketing) are only allowed to have a maximum of 10 accounts in total, whatever is defined at the sub-levels.

You can use the Details buttons inside the tree view to review the quota chain which is applying on a context.

- `LDAP Defaults`: List of default attribute values which will be auto-filled when creating or extending objects inside the target subtree.

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).

Figure 46. AdminRole Assigned Group

8.1 Basic 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 (ie. 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.

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.

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 TSL.
- `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.

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.

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 any 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.

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 Trust Domains

Trusts are special Domains which do not correspond to a set of local LDAP users but a domain on a remote WebADM installation. The Trust system works like an authentication proxy for remote domains (within a trusted organization) and maps a local virtual Domain name to a remote Domain on another WebADM server.

Trust work only with Web Services. The local Web Service will forward the incoming SOAP request to the trusted system through secure communication.

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

A WebADM Trust object supports the following settings:

- `Remote Server Address`: This is the server name (hostname) for the remote services.
- `Remote Server SOAP Port`: This is the remote SOAP port (typically 8080 or 8443).
- `Use SSL Connection`: If enabled, connection will use SSL (port 8443).
- `Trusted CA Certification File`: This is the file containing the remote CA certificate. If set only the certificates issued by this CA will be trusted for remote server authentication.
- `Trusted Certificate CN`: Mandatory certificate common name required for remote server authentication. Trusted certificates with non-matching CN will be refused.
- `Remote Domain Name`: If the remote domain name is not the same as the local Trust Domain name, you can specify the remote domain.

The user access policy settings (Address, Hours and Applications' restrictions) are similar to the WebADM LDAP Domains' settings described previously. Please see the previous section for details.

Figure 58. Trust Domain Settings

12. 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.

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.OTPType=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.

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.

13. 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.

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.

13.1 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

13.2 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

13.3 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

13.4 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

13.5 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

14. 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`).

15. 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*.

Figure 68. Inventory Database

16. 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.

17. Managing User 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.

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.

Figure 70. User Certificate Creation

WebADM can use an external CA for signing the user certificates. You can set the Sign Mode to Ext in your OptionSet for this purpose. Note that with the OptionSets, you can also specify a different signing method for different parts of your LDAP tree. The Rsign internal signing mode is required by some WebApps (such as SelfDesk) and Web Services which need an automated certificates signing system.

When you configure the user certificate as log in method (`auth_mode` in `conf/webadm.conf`) for the WebADM Administrator Portal authentication, you can enable the certificate revocation system in the OptionSet.

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.

17.1 RSign Internal PKI

WebADM RSign is designed for the WebADM application only. It provides the minimal features needed to automatically and immediately deliver administrator's and user's client certificates without requiring a human process.

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.

17.2 RSignd 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.

17.3 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.

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.

Figure 73. User Settings Editor

19. Using the Manager Interface

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 Web App 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 and PKI server are very high-performance systems, multithreaded and written in C. Those components do not call external network services such as LDAP or SQL servers and can also handle very high numbers of requests.

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 (ie. 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.

24. Common Issues

OpenLDAP server may forbid adding the inetOrgPerson object class to an existing user. In that case, you may need to re-create your users in WebADM or remove the inetOrgPerson object class from the user object specifications in the `conf/objects.xml` file.

With Microsoft ActiveDirectory, you must add the WebADM proxy user to the Domain Admins group. Be sure have the SSL enabled and to respect the domain password complexity policy for the proxy user password and user passwords. Else, WebADM will not be able to create users or set passwords.

With MySQL and MariaDB and WebADM 1.4.2 you might get connection errors with "Host 'hostname' is blocked" because of many connection errors; unblock with 'mysqladmin flush-hosts'. This is due to the socket polling system of WebADM Watchd which checks the database connection is available by opening TCP sockets every 10 seconds. Recent versions of MySQL counts these socket checks as failed connections and also blacklists the WebADM servers after 100 checks. The workaround simply consists of re-configuring the failed connections' limit to the maximum value in MySQL.

On 32bit MySQL servers, set max_connect_errors=4294967295. On 64bit MySQL servers, set max_connect_errors=18446744073709551615.

This issue is fixed in WebADM v1.4.3.

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.
# 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

# 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 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", "dynamicGroup",
"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"

# ignore some AD attributes
ignored_attrs "ntsecuritydescriptor", "objectcategory", "objectsid", "badpasswordtime",
\
              "badpwdcount", "lastlogoff", "lastlogon", "logoncount",
"lastlogontimestamp", "pwdlastset", "primarygroupid", "samaccounttype"

# Find below the LDAP containers required by WebADM.
# Change the container's DN to fit your ldap tree base.
# WebADM AdminRoles container
adminroles_container "dc=AdminRoles,dc=WebADM"
# WebADM Optionsets container
optionsets_container  "dc=OptionSets,dc=WebADM"
# WebApp configurations container
webapps_container "dc=WebApps,dc=WebADM"
# WebSrv configurations container
websrvs_container "dc=WebSrvs,dc=WebADM"
# Mount points container
```

```
# Mount points container
mountpoints_container "dc=MountPoints,dc=WebADM"
# Domain and Trusts container
domains_container "dc=Domains,dc=WebADM"
# Clients container
clients_container "dc=Clients,dc=WebADM"

# With MS Active Directory use the following settings instead of the previous ones
# Note: Replace dc=mydomain,dc=com with your AD domain DN
#adminroles_container "cn=AdminRoles,cn=WebADM,dc=mydomain,dc=com"
#optionsets_container "cn=OptionSets,cn=WebADM,dc=mydomain,dc=com"
#webapps_container "cn=WebApps,cn=WebADM,dc=mydomain,dc=com"
#websrvs_container "cn=WebSrvs,cn=WebADM,dc=mydomain,dc=com"
#mountpoints_container "cn=Mountpoints,cn=WebADM,dc=mydomain,dc=com"
#domains_container "cn=Domains,cn=WebADM,dc=mydomain,dc=com"
#clients_container "cn=Clients,cn=WebADM,dc=mydomain,dc=com"

# 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  "cq19TEHgHLQuO09DXzjOw30rrQDLsPkT3NiL6l3BH2w="

# Hardware Cryptographic Module
# Only Yubico YubiHSM is currently supported for WebADM hardware encryption.
# Up to 8 HSM modules can be concurrently attached to the server.
#hsm_model YubiHSM
#hsm_keyid 1

# 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
```

```
# 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 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_format Default
#log_syslog No
#syslog_facility LOG_USER
#syslog_format CEF

# Alerts are always recorded to the SQL Alert log. Additionally, when alert_email
# is defined, the alerts are also sent by email to the configured recipient(s).
#alert_email “me@mydomain.com”

# 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 create the
# HTTP_X_FORWARDED_FOR and HTTP_X_FORWARDED_HOST headers.
#reverse_proxies "192.168.0.100", "192.168.0.101"
# If you use WebADM Publishing Proxy (WAProxy) for publishing applications 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!
```

```
#waproxy_proxies "192.168.0.102"

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

# WebApps theme
# Comment the following line to disable the default theme.
webapps_theme "default"

# End-user messages
# The following variables are available: %USERNAME%, %USERDN%, %USERID%, %DOMAIN%,
%APPNAME%
# Additional variables are available depending on the context: %APPID%, %TIMEOUT%,
%EXPIRES%
unlock_subject "Unlocked access to %APPNAME%"
unlock_message "Hello %USERNAME%,\r\n\r\nYou have a one-time access to the
%APPNAME%.\r\nYour access will automatically expire %EXPIRES%."

# Personalization options
# You can customize your organization name, logo file and website URL.
# The logo file must be PNG image with size 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 2000
#default_portal Admin
#ldap_uidcase No
```

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.
```

```
will automatically failover to the next configured server.

At least one LDAP 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_cert: Trusted CA for SSL and TLS
-->

<LdapServer name="LDAP Server"
        host="localhost"
        port="389"
        encryption="TLS" />

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

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

Allowed LDAP parameters are:
 - type: MySQL, 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
 - tnsname: Oracle TNS name (Oracle only)

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 wright 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" />

<!--
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.
–>

<SessionServer name="Session Server"
        host="localhost"
        port="4000"
        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"
        ca_file="" />

<!--
WebADM supports YubiHSM connected locally or the network-based
RCDevs HSM Server (HSMHub). With RCDevs HSMHub, the HSM server
connection parameters must be set below.
-->

<!--
<HsmServer name="HSM Server"
        host="remotehost"
        port="6000"
        secret="secret"
        ca_file="" />
-->

<!--
You need to configure RCDevs Push server(s) in order to use
TiQR Push or RCDevs Mobile Authenticator with Push login.
If you have an Enterprise license then you don't need to
register an RCDevs push account (keep user and password empty).
-->

<!--
```

```
<PushServer name="Push Server"
        host="push.rcdevs.com"
        port="7000"
        user=""
        password=""
        ca_file="" />
-->

<!--
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 /usb/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
cert_validity 365

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

# In proxy mode, you do not need the following settings
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
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

#
# Proxy settings
#
# If the server is a rsignd proxy, use "proxy yes" here.
# Specify the real rsignd server IP and service port number.
proxy_mode no
#server_name xxx.xxx.xxx.xxx
#server_port 5000

#
# 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 server(s) must be defined too.

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
)
```