OpenMDM® 5 Rights and Roles

Version 1.0.0

Table of Contents

The purpose of this document is to define the structure of Rights & Roles features in the context of the OpenMDM® 5 architecture.

Chapter 1. License

Eclipse Public License - v 1.0

THE ACCOMPANYING DOCUMENT IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT’S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

a. in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

b. in the case of each subsequent Contributor:

   i. changes to the Program, and

   ii. additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor’s behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

a. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form. .Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

b. Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient’s responsibility to acquire that license before distributing the Program.

c. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

a. it complies with the terms and conditions of this Agreement; and

b. its license agreement:

   i. effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

   ii. effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

   iii. states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

   iv. states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

a. it must be made available under this Agreement; and

b. a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor’s responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement , including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR

DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Chapter 2. Introduction

The OpenMDM® Working Group recognized the need to provide user authentication and user authorization as part of the OpenMDM® 5 framework. An initial workshop on user authentication and authorization in the context of OpenMDM® was held in April 2016. Its results are documented at https://openmdm.atlassian.net/browse/ORGA-98.

Chapter 3. Objectives

The following objectives must be fulfilled by any possible implementation of the Right & Roles concept.

3.1. Authentication

- Integration with existing enterprise authentication mechanisms and standards (such as PKI and Kerberos for example).

- When LDAP or Active Directory are available, use the information provided by those systems instead of maintaining independent copies of the information.

- When single sign-on mechanisms are available (such as Kerberos), OpenMDM® must integrate seamlessly, without forcing the user to present their credentials once again.

- When the authenticated entity must be passed form OpenMDM® to external systems then established standards for identity propagation must be used, such as CSIv2/SAS, SAML, JWT (where appropriate).

- The identity (trace) of the creator of an OpenMDM® object must be stored in relationship with the object. Such identity must be stored in a form that allows reconstruction even when the identity has been removed from the central authentication database (i.e, the employee has left the company).

3.2. Authorization

- Support a base data container (known as “data pool”) on which roles apply to all OpenMDM® objects related to it (such as Measurements, Tests, etc).

- Access rights must be assignable per data pool. If the data pool is physically spread over multiple servers then those rights must be honored by all servers.

- Access rights must be primarily handled by roles, allowing the concepts of user groups. In exceptional cases access rights may be assigned to individuals.

- Access rights must be stored along with the data entities to which they pertain.

- Access checks must be performed at the data server level. (This is to ensure that access checks cannot be bypassed by login into the data server directly).

Chapter 4. Design Guidelines

Data must be protected at the source, i.e., either directly at the data store level or within a service layer that completely encapsulates the data store and cannot be bypassed.

Data protection must work application-independent, i.e., must produce the same results irrespective of the specific application used to access the data. If access to a particular data item is permitted, it should be visible in every application designed to work with that data; if access is not permitted, the data item must not be exposed through any application.

Any action on protected data must be attributable to a specific entity ("principal") that can be held responsible for the action. The principal can be a human or (in the case of batch processing) a program.

Chapter 5. Implementation

There are a couple of ways on how Authentication and Authorization may be handled in the context of an OpenMDM® system while making sure the previously stated objectives are fulfilled.

5.1. Authentication

Use existing authentication system already available at the deploy location (such as LDAP, AD, Kerberos) as long as the OpenMDM® requires network access and remote data access.

In cases where the OpenMDM® application is deployed on a standalone fashion or used in offline mode then the local authentication mechanism (such as operating system login) is enough.

5.2. Authorization

In cases where the OpenMDM® application is deployed on a standalone fashion or used in offline mode then the user may have access rights to every operation the application provides. However access rights must be checked when the application connects with a remote server or attempts data transfer. For all other cases authorization may be handled in 3 different approaches

5.2.1. Native

Defines an OpenMDM® specific Access Rights API and implementation of that API (i.e., of administration functions and runtime access checking) within OpenMDM® as a library or infrastructure component. This means that all security checking is carried out at the OpenMDM® level and not in the underlying data server (e.g., the ODS server). OpenMDM® would access the data server with superuser privileges to bypass all checking within the server itself and would then carry out its own access checks according to the access rights defined at the OpenMDM® level.

PROS

- Clean room design and implementation.
- Integration with existing authentication mechanism need to be implemented once at the OpenMDM® level, not at every data backend level.

CONS

- Major undertaking with all stakeholders in order to design the model (like it happened with ODS for example).
- Direct access to the data level is forbidden, thus all existing tools must migrate to OpenMDM® as soon as possible.
- Access rights cannot be used for optimized queries at the data level.

5.2.2. Hybrid

Define an OpenMDM® specific Access Rights API, but implementation of that API is delegated to the data server. This means that OpenMDM® would provide a common API to view and administrate access rights, but each server adapter would map calls to this API into calls to the security API of the underlying data server. For example, an ODS adapter would map the assignment of a particular access right to a particular group via the OpenMDM® API into the creation of a corresponding ACL entry in the ODS server. The actual access checks would then be performed by the data server.

PROS

- Clean room design.
- Data protection occurs at the source (data level) thus OpenMDM® and non OpenMDM® applications can rely on it.

CONS

- Mapping of security model to every data backend (ODS, PAK Cloud, etc). This include interfaces with existing enterprise authentication mechanisms.
- The API must be designed as the least common denominator between existing data backends. Backend specific extensions must be designed and implemented.

5.2.3. Delegate

Complete delegation to the data server. This means that OpenMDM® provides neither an API nor an implementation for access rights management, but simply relies on the access checks carried out by the data server. In other words, access rights are administrated and checked at the data server level exclusively; OpenMDM® simply passes the current user identity along with each server call and relies on the data server to expose only the data items the current user is entitled to see.

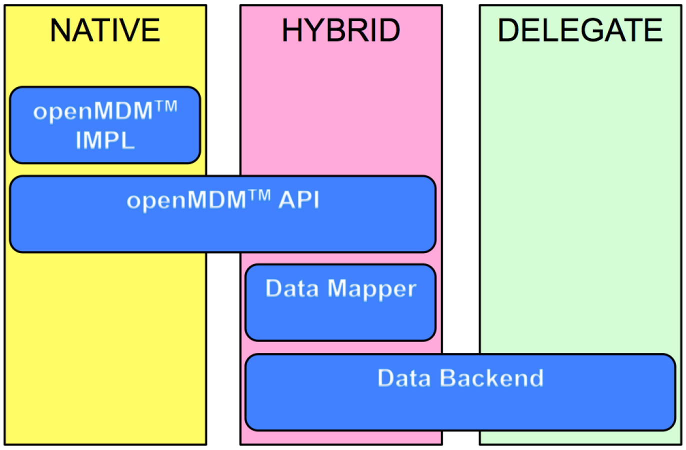
PROS

- Data protection occurs at the source (data level) thus OpenMDM® and non OpenMDM® applications can rely on it.

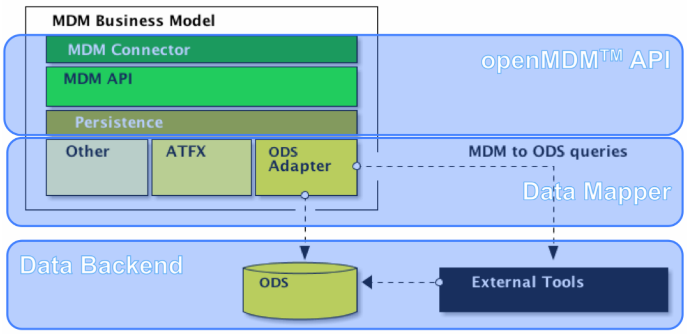
CONS

- Absence of a coherent OpenMDM® security model as each data backend implements security in its own way.
- Integration with existing enterprise authentication mechanisms must be executed by each data backend.

These approaches can be mapped to the OpenMDM® architecture in the following way



The OpenMDM® architecture has been designed in such a way that components can be built as an aggregation of base services and other OpenMDM® components, allowing for behavior reuse and interchangeable implementations among OEMs and providers. It's worth remarking that the OpenMDM® architecture allows OEMs to choose the particular data backend that better suits their needs. For this reason, the architecture defines a layered structure that makes it easy to plug in an specific data backend such as an ODS provider. For simplicity, the rest of this document will use ODS as data backend.



Chapter 6. Recommendation

The **Delegate** approach is the path of least resistance, as it only requires propagation of the user identity to the data server on each call. Starting with Delegate does not exclude the Hybrid approach, as delegation (via mapping) must happen as well. Only the Native approach requires a separate design and implementation. It also requires a major undertaking by all stakeholders to agree on design. This time and cost-wise (in the short term) Delegate is the preferred approach on which Hybrid can be built later. Native makes sense when all stakeholders and participants can move to OpenMDM® together. The OpenMDM® API requires two changes for this proposal to work:

- A login request that can be forwarded to the backend. The result of this request is either an error or an identity token.
- All other requests (such as queries, mutators) must include the identity token as part of its arguments.

In the case of a search request, the identity token could be used to reconstruct the real identity attached to an OpenMDM® object, allowing faster and “native” searches at the backend level, instead of a two-pass search & filter alternative at the OpenMDM® API level.

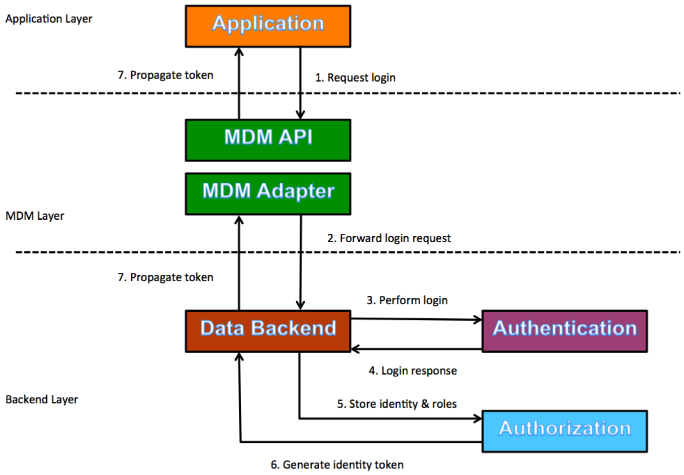


Figure 1. Authentication in Delegate mode

Authentication Data Flow

1. Application issues a login request.
2. The MDM API forwards the login request to the MDM Adapter.
3. The Data backend (ODS for example) receives the request and forwards it to the Authorization module (LDAP/AD/etc).
4. Authorization module either accepts the login credentials and returns an identity with roles or fails.
5. If login was successful then identity and roles are stored temporarily (think of a session).

6. An authentication token is generated. The token must be used in all other incoming requests in order to grant access to operations.

7. The token is forwarded all the way to the application.

Tools such as Matlab skip the MDM layer of course. It’s likely that some organizations already have a token based authentication system in place, in which case said token may be reused for the Delegate approach. However it’s very important that the data required by the authorization module be made available at the backend layer. This means further integration must be built such that the data is added to the authorization module at the right time, also removed/cleared when there’s no longer a need for it. This additional integration must be built custom made for each target backend and authorization provider.

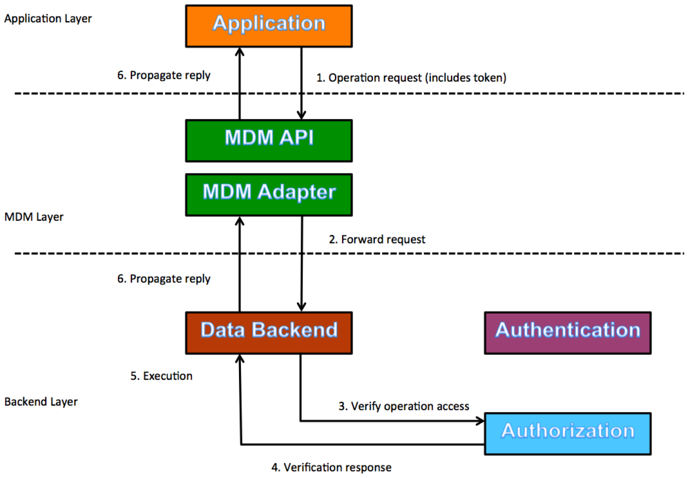


Figure 2. Authorization in Delegate mode

Authorization Data Flow

1. Application issues an operation request (store data for example). This request contains the authenticated identity token.

2. The MDM API forwards the operation request to the MDM Adapter.

3. The Data backend (ODS for example) validates the identity token and checks access rights.

4. Valid token and correct access rights grants green light to the operation to continue. Failures result in denied access; possible authentication workflow if no token or expired token.

5. Execute the requested operation.

6. Propagate results to the application.

6.3. Comparison with Hybrid

The Hybrid approach can reuse code and components built for the Delegate solution. Adds a new API on top of the OpenMDM® API to cache, validate, and verify the security token and roles & rights. The following images depict the changes required between Delegate and Hybrid.

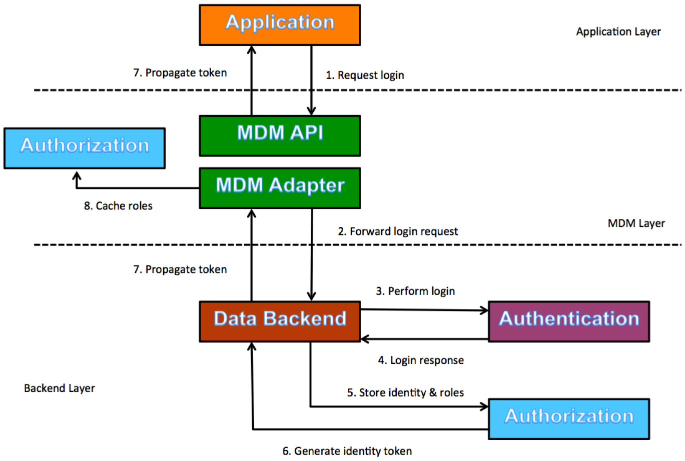


Figure 3. Authentication in Hybrid mode

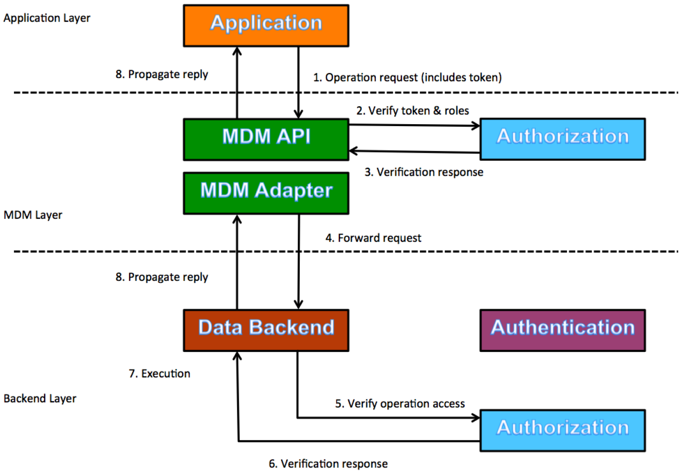


Figure 4. Authorization in Hybrid mode

In terms of API changes OpenMDM® API requires new types to handle Identities and Roles, alongside a Verification module/service. All requests (except login) must include an identity token after authentication is successful. All requests (except login) must be handled by the Verification module to check access rights before sending down the stack through the MDM adapter.