|  |
| --- |
| oneM2M-Logo |

|  |  |
| --- | --- |
| **oneM2M**  **Technical Report** | |
| Document Number | TR-0048-V0.0.3 |
| Document Name: | oneM2M App-ID Registry Function |
| Date: | 2017-Dec-04 |
| Abstract: | Technical Report of oneM2M End to App-ID Registry Function to support the oneM2M security and enrolment procedures. |

This Specification is provided for future development work within oneM2M only. The Partners accept no liability for any use of this Specification.

The present document has not been subject to any approval process by the oneM2M Partners Type 1. Published oneM2M specifications and reports for implementation should be obtained via the oneM2M Partners' Publications Offices.

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Copyright Notification

No part of this document may be reproduced, in an electronic retrieval system or otherwise, except as authorized by written permission.

The copyright and the foregoing restriction extend to reproduction in all media.

© 2016, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC).

All rights reserved.

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT PARTNER FOR THIS DOCUMENT, WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

# Contents

Contents 3

1 Scope 5

2 References 5

2.1 Normative references 5

2.2 Informative references 5

3 Definitions, symbols, abbreviations and acronyms 5

3.1 Definitions 5

3.2 Symbols 6

3.3 Abbreviations 6

3.4 Acronyms 6

4 Conventions 6

5 Introduction 6

6 Use Cases 8

6.1 Use Case of 1: Verify IoT Application identity and integrity 8

6.2 Use Case 2: Allow Certified IoT applications 9

6.2.1 Description 9

6.3 Use Case 3: Registration Enrolment using the App-ID Metadata 9

6.3.1 Description 9

6.4 Use Case 4: IoT Data model mapping to oneM2M ontology 10

6.4.1 Description 10

6.5 Use Case 5: IoT application Certificate – Trusted Root 11

6.5.1 Description 11

6.6 Use Case 6: Revoke App-ID or AE-ID 13

6.6.1 Description 13

6.7 Actors 13

6.8 Pre-conditions 13

6.9 Normal Flow 14

6.10 Revocation - Normal Flow 15

6.11 Potential requirements 16

7 Candidate Architecture 17

7.1 Candidate Architecture for App-ID Registry Function 17

7.1.1 OPTION #1 – Standalone App-ID Registry function (ARF) 17

7.1.2 OPTION #2 – App-ID Registry Function within MEF 17

7.2 Architecture Proposal For Registration Enrolment using the App-ID Metadata 18

7.2.1 Functionality Requirements 18

7.2.2 Architecture Option #1 18

7.2.1.1 Nodes 20

7.2.1.2 Reference Points 20

7.2.3 Architecture Option #2 21

7.2.3.1 Nodes 21

7.2.3.2 Reference Points 22

7.3.2 Architecture 23

7.3.2.1 Nodes 25

7.3.2.2 Reference Points 25

7.4 Architecture Proposal For functionality X 25

7.4.1 Functionality X Requirements 25

7.4.2 Architecture 25

7.4.2.1 Nodes 25

7.3.2.2 Reference Points 25

8 App-ID Registry Function Metadata 26

8.1 App-ID Registry Function – Metadata related Requirements 26

8.2 App-ID Metadata 26

9 Available Options 28

9.1 Review of Existing Technology for use in the functionality 28

9.1.1 Review of Existing Technology N 28

9.1.1.1 Introduction to Technology N 28

9.2 A Solution for providing Functionality A 28

9.2.1 General procedure for providing functionality A 28

10 Release X Function Rationale 29

10.1 Overview of Release X Features 29

10.2 Release X Function 29

10.2.1 X Function Overview 29

10.2.2 X Functional Architecture 29

11 Conclusions and recommendations 29

Annex A: Problem Statement for needing App-ID Registry Function 30

A.1 Introduction 30

History 30

# 1 Scope

The present document provides options and analyses for an App-ID Registry Function to support oneM2M enrolment and security features, supporting mechanisms to provide end-to-end security for oneM2M.

The scope of this technical report includes use cases, threat analyses, high level architecture, generic requirements, available options, evaluation of options, and detailed procedures for an App-ID Registry Function within the oneM2M security authentication and enrolment procedures.

This technical report will provide use case scenarios for how the App-ID registry can be used as an integral component of the oneM2M system architecture providing App-ID registry based functions / services.

# 2 References

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

## 2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

## 2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] oneM2M TR-0004 Definitions and Acronyms

[i.2] W3C Recommendation “Canonical XML Version 1.0”, 2001, <http://www.w3.org/TR/xml-c14n>

[i.3] IETF RFC 7165: “Use Cases and Requirements for JSON Object Signing and Encryption (JOSE)”

[i.4] IETF RFC 5166: “An Interface and Algorithms for Authenticated Encryption”, 2008

[i.5] oneM2M drafting rules (draft)

[i.6] oneM2M TS-0001 Functional Architecture

[i.7] oneM2M TS-0002 Requirements

[i.8] oneM2M TS-0003 Security Solutions

[i.9] oneM2M TS-0004 Service Layer Core Protocol Specification

# 3 Definitions, symbols, abbreviations and acronyms

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in [i.1] and the following apply:

NOTE: This may contain additional information.

## 3.2 Symbols

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

|| Concatenation

## 3.3 Abbreviations

For the purposes of the present document, the [following] abbreviations [given in ... and the following] apply:

TBD

## 3.4 Acronyms

For the purposes of the present document, the abbreviations [i.1] apply:

TBD

# 4 Conventions

The key words “Shall”, ”Shall not”, “May”, ”Need not”, “Should”, ”Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.5]

# 5 Introduction

In today’s rapidly growing IoT environment, authenticity and data security problems are rampant and pose great challenges. These include unsecured supply chains for IoT devices, no mechanism to manage unknown IoT devices, as well as BYOD (Bring Your Own Device) consumer IoT services that enable connections of devices outside of direct control of the IoT platform or service providers.

Though IoT devices can support a wide range of security mechanisms, including X.509 Certs, pre-shared keys, raw public keys, many solutions use no security at all. From toasters to baby monitors, IoT devices have shown vulnerability to cyberattacks.

Furthermore, in the current environment, IoT security is highly fragmented and characterized by vertical standards, proprietary technical implementations and weak security that creates silos and restricts interoperability. As such, a reliable mechanism to validate identity integrity for a connecting IoT application is critical to securing IoT services and the resulting trust that interoperability depends upon.

For many IoT applications that support critical infrastructures such as healthcare, the smart city and emergency services, device compromise can be a significant concern, with potentially catastrophic consequences.

**App-ID Registry**

The App-ID Registry enables applications to identify themselves in a consistent, standards-based way to the oneM2M service layer. It provides a guaranteed means for uniquely identifying each oneM2M application and device. This enables any IoT application from any developer to communicate and share data with any IoT device. In addition, it offers trusted identity and authentication of IoT application data.

It is particularly valuable for organizations that do not operate within the same networks or platforms and which could not easily communicate with one another. The App-ID Registry helps to bridge these silos enabling a much broader addressable market opportunity for application developers, device maker, and service providers.

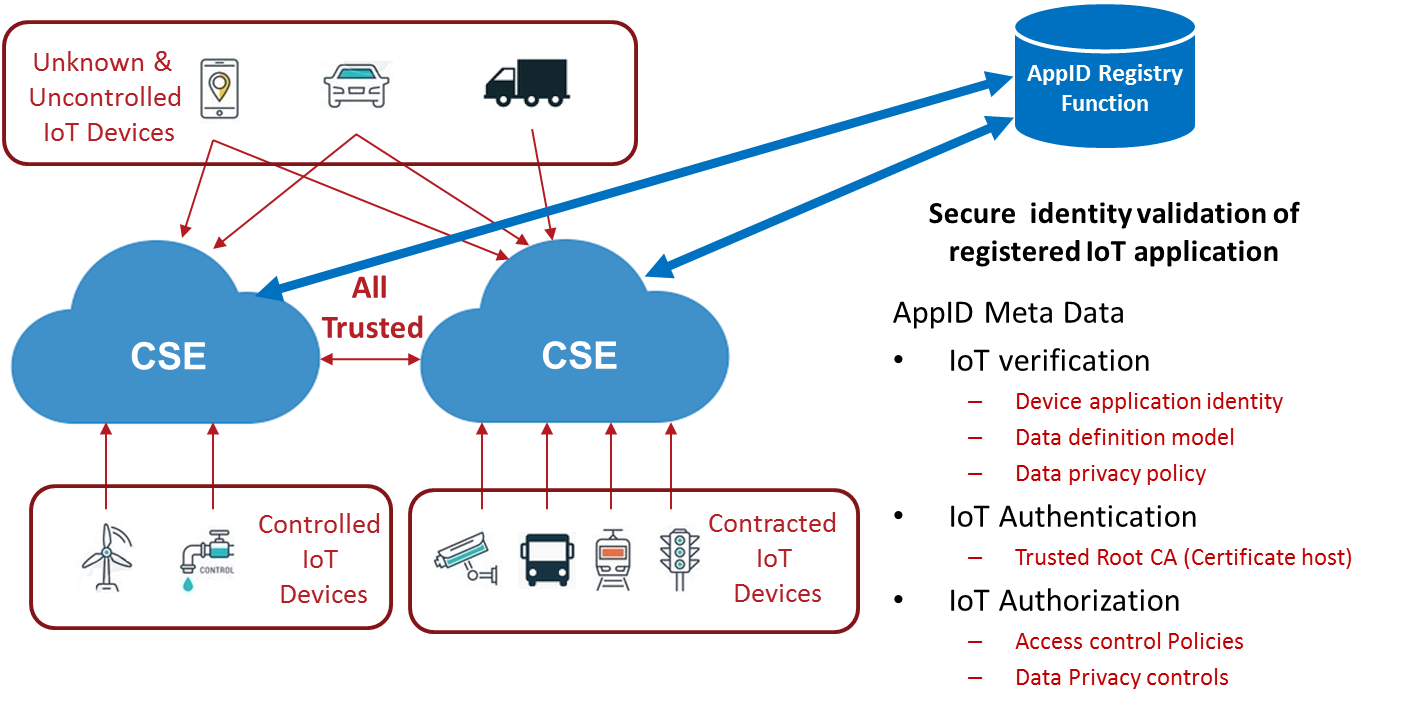
For example, a smart city will likely use multiple suppliers for its IoT devices and applications as well as support multiple users of this data. It could also support “visiting devices” from tourists and business travellers, making it very difficult for these devices to securely communicate. Data interoperability is critical for successful implementation of many smart city solutions – such as transportation, public safety, event management and government services, as a few examples.

Another example where interoperability is critical to success is the smart home. Many companies engaging in this space only develop products and applications for one or two device categories – such as smart locks or lighting controls. For service providers to offer a complete smart-home solution, it will be necessary for multiple solutions to work together seamlessly. Partnering becomes much easier once developers open up their interfaces and use the App-ID Registry to gain visibility in the smart-home ecosystem.

The App-ID Registry offers numerous benefits across the entire IoT ecosystem.

For device and application vendors, the Registry provides the ability for IoT devices to be uniquely identified and authenticated so as to be broadly adopted by any IoT service provider, increasing the addressable market opportunity. The App-ID Registry provides metadata regarding the characterization of the IoT device and the format of the data it produces, thus interoperable for integration with any appropriate application. All this enable applications to be compatible with a greater range of devices, thereby improving cost effectiveness, while enhancing the rate of innovation. Finally, compliance test and certification of IoT applications, increase buyer confidence and brand value.

* For IoT platform and system integrators, the App-ID Registry helps to streamline the onboarding process and allows integration with broader range of IoT devices, while reducing the cost of ongoing management.
* For IoT network operators and service providers, the App-ID Registry encourages open yet trusted access which results in broader adoption for a larger range of IoT devices and their data to enable more innovation and the resulting revenue possibilities.
* For consumers and other end users, the App-ID Registry makes it easier to participate in services in an IoT BYOD environment, and supports privacy controls over data use.



**Figure 5: App-ID Registry Function Secure Identity Validation**

As the number of devices, applications and developers in the IoT market continues to grow, the App-ID Registry can confirm that IoT devices and their application identities are both unique and discoverable with levels of trust that enable all this to be managed at scale.

For the IoT industry, the App-ID Registry lowers the cost of implementation, integration and ongoing management of IoT applications and devices; and enables a broader inclusion for IoT device manufacturers and consumers. Perhaps most importantly, IoT applications and the data they produce can be trusted, limiting IoT security vulnerabilities and maximizing service innovation through increased interoperability that is also secure.

# 6 Use Cases

## 6.1 Use Case of 1: Verify IoT Application identity and integrity

IoT applications can enter into the supply chain via many routes. With the best intentions it is not possible to 100% secure the control of the supply chain though manufacturers, distributors, resellers, 3rd party contractors and consumers.

Many IoT service are required to provide open access for unknown IoT applications to connect, for example smart homes, cities, healthcare. The economies of these services are built upon consumers and businesses being able to acquire their own IoT applications and connect them with these smart communities.

Even when IoT systems are vertically integrated, such as industrial controls, factories, utilities etc, the IoT service provider will find it difficult to control the supply chain and cannot guarantee 100% that all the IoT applications are authentic.

Where this happens at scale for example 1000’s IoT sensors with an industrial deployment or a smart city needing to authenticate various brands makes and models of connected vehicles. Scale exacerbates the problem for supply chain controls.

**Using App-ID Registry Function to provide IoT identity and integrity checking.**

A Service provider Infrastructure can verify the identity of a connecting IoT application by connecting with the App-ID Registry Function. A SP is able to query the App-ID Registry Function if the App-ID is registered and if so collect the metadata for a presented IoT application’s (AE-ID/App-ID). The App-ID metadata can contain information to enable the SP to verify if the connecting IoT application. The App-ID metadata can contain the following attributes to allow the SP be informed to make decisions how to allow the IoT application connect, if at all :-

* **IoT application Type:** Information regarding the type of application, for example thermostat, vehicle telemetry, or fan control.
* **IoT application Class:** Details for the class of use: consumer, industrial, or critical infrastructure. The~~se~~ class type will have a bearing on how the data and security of the IoT application is enrolled onto the SP.
* **Security capability**: The security capabilities of the IoT application. This will indicate the security capabilities of the IoT application, authentication, as well as the ability for the IoT application to secure a private key in a trust enclave.
* **Authentication type:** The mechanism used to authenticate the IoT Application. For example, if using a PKI certificate, who is the root of trust.
* **Communications class:** The profile for the data generated by the application. For example, streamed data, periodic burst of data, or intermittent bursts of data.
* **Data Model:** The data definition for the application, what is projected, and the actions that can be performed. The application data model and mapping to oneM2M ontology.

Using the App-ID metadata provided by the App-ID Registry Function, the SP can verify the identity and authenticity of the connecting IoT application. Also it can assign the right resources for the connecting IoT application

This functionality will allow any registered App-ID to provide an application profile in form of metadata that can be used by a SP to identify and verify that the connecting IoT application is representing itself consistently with the characterization presented by the metadata.

The role of the App-ID Registry Function is not to enforce the policy of the SP, but for the SP to be informed through the App-ID metadata profile to automate the authentication and enrolment process.

## 6.2 Use Case 2: Allow Certified IoT applications

### 6.2.1 Description

Some specific IoT services rely on strict industry certification for the use of any IoT application. The IoT application needs to comply with specific industry regulation, technical specification, consumer rights and industry specific security policies. For example, a power station may require strict environmental and security compliance for its use.

To this end, test and certification bodies provide services to ensure that an IoT application are conformant with the appropriate specifications for compliance.

Today a Service Provider has no context of an unknown IoT application being compliant with any specific certifications or not. I.e. is a heart rate monitor or blood pressure monitor certified and should the data be allowed to be passed into the client’s health record. The consequence of such could be damaging not only to the patient, the credibility of the health care provider and IoT service provider

**Using App-ID Registry Function to verify the IoT application is certified for use.**

The service provider infrastructure can verify, with the App-ID Registry Function, that the identityof a connecting IoT application is certified for use and complies with specifications. The SP is able to query the App-ID Registry Function if the App-ID is registered and if so collect the metadata for a presented IoT application’s (AE-ID/App-ID). The App-ID metadata can contain information to enable the SP to verify if the connecting IoT application is certified for the specific use. The App-ID metadata can contain the following attributes to allow the SP be informed regarding the certification status and to make decisions how to allow the IoT application connect, if at all :-

* **Certification body**: the test and certification body that has verified the App-ID identity and the metadata profile.
* **Certification compliance**: List of the compliance functionality that is certified by the certification body. Not all features of the certification process may be compatible with the IoT application. It is possible only a sub set of features are compliant with the certification process.

Using the App-ID metadata provided by the App-ID Registry Function, the SP infrastructure can verify the App-ID certification compliance of the connecting IoT application and if it should be allowed to connect or provide any data to the IoT service.

The role of the App-ID Registry Function is not to enforce the policy of the SP infrastructure, but for the SP infrastructure to be informed through the App-ID metadata profile to automate the authentication and enrolment process.

## 6.3 Use Case 3: Registration Enrolment using the App-ID Metadata

### 6.3.1 Description

Where the connecting IoT application (AE-ID/App-ID ) is unknown to the oneM2M system, the administrator of the system, shall configure the IoT application identity service subscription rule <serviceSubscribedAppRule> to allow the IoT application to connect with the Service Provider infrastructure and provide its data.

The <serviceSubscribedAppRule> resource represents a rule that defines allowed Role-ID, App-ID and AE-ID combinations that are acceptable for registering an AE on a service provider’s infrastructure.



**Figure 6.3.1: Service Subscription App Rule**

The rule contained in a <serviceSubscribedAppRule> resource defines a mapping between:

a) one or more Credential-ID(s); and

b) combinations of one or more Role-ID(s), one or more App-ID(s) and one or more AE-ID(s) which are allowed to be used for registering AE(s) that issued a registration request via a Security Association established with the credentials associated with the Credential-ID(s)

Today if the Service Provider has no prior knowledge of the IoT application (AE-ID/App-ID ) then the IoT application will not authenticate and the administrator would need to manually configure the <serviceSubscribedAppRule>. This presents issues for both scale through mass enrolment and/or enrolment of unknown IoT applications.

**Using App-ID Registry Function to auto enroll <serviceSubscribedAppRule**>

By connecting with the App-ID Registry Function, the service provider infrastructure can query the metadata for a presented IoT application’s (AE-ID/App-ID ) to auto-populate the service subscription.

Where the AE-ID/App-ID are unknown to the SP infrastructure. Using the App-ID Registry Function will enable the <serviceSubscribedAppRule> to be auto-populated from the metadata provided from the App-ID Registry Function.

## 6.4 Use Case 4: IoT Data model mapping to oneM2M ontology

### 6.4.1 Description

IoT applications of the same type and functionality may all have different data models to represent their information with the Service Providers infrastructure. For example a light switch from different manufacturers may present on/off as 1/0 (One/Zero) or true/false, or some other representation.. Although oneM2M has defined data models/ontologies for specific IoT application class types, this may not be followed be the IoT application developer and or could contain proprietary extensions to differentiate itself in the market.

More so when the IoT is not natively oneM2M and is connected via an interworking function. The data may not have any representations which match the defined oneM2M ontology definitions.

Although it is possible to do this in a manual process, it would require prior knowledge of the connecting IoT application type/model and possibly the software version.

**Using App-ID Registry Function to automate data model mapping to oneM2M ontologies**

App-ID metadata provides a description of the specific IoT application data model definition, which describes the mapping to a compatible oneM2M ontology. This will enable a Service Provider to automate the process for enrolling the IoT application. The SP will be able to manage the data it produces and any interactions to control its functionality. The App-ID metadata can provide the following specific information:

* **Data Model:** The data definition for the application, what is projected, and the actions that can be performed. The application data model and mapping to oneM2M ontology.
* **Data privacy:** The data privacy asserted for this App-ID. The generic data privacy model for the application identity. For example, the IoT application is a blood pressure monitor and the data can only be provided to the client’s electronic health record. The data cannot be shared, data mined, or resold by the IoT service provider

By connecting with the App-ID Registry Function, a Service Providers infrastructure can query the metadata for a presented IoT application’s (AE-ID/App-ID) and retrieve a data model definition to both verify the data being used and to manage the permissible use.

**The benefits to the eco-system are:**

– Providing SEMANTIC interoperability

• Annotate M2M data with information, describing e.g.

– Name of the data (this could contain namespace / ontology).

– Relation to other M2M data.

• Abstraction from specific technologies

– Support of Data Brokering / Analytics / Big Data

• Semantic Discovery

– Can be supplemented with additional context information

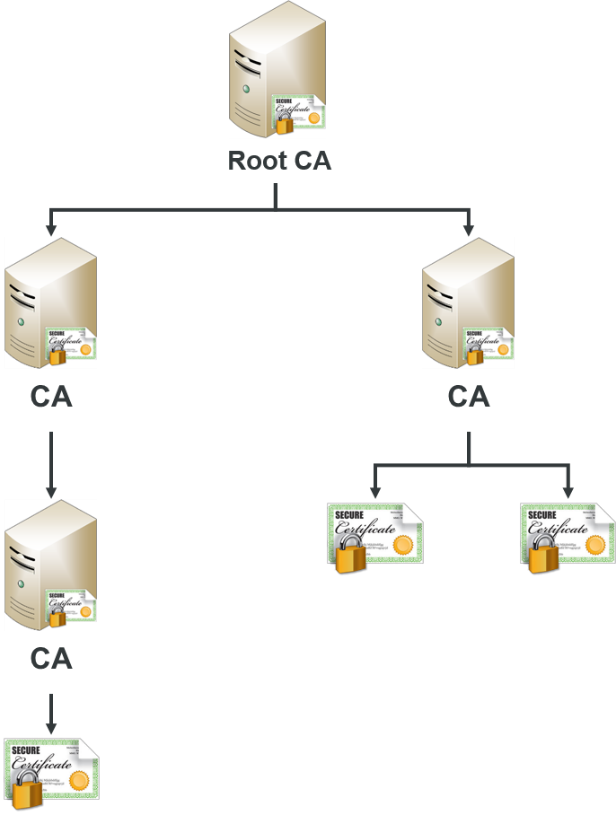
• Data brokering (advertising available data / finding relevant data)

– Support of Big Data Analytics

## 6.5 Use Case 5: IoT application Certificate – Trusted Root

### 6.5.1 Description

A Certification Authority (CA) may either issue CA certificates or end entity certificates. A Root CA has a self-signed CA certificate and issues CA certificates to subordinate CAs. Trust in an end entity certificate requires trusting the Root CA.



**Figure 6.5.1: Certificate Hierarchy**

Digital certificates are stored in a certificate store. Major operating system (OS) vendors, Microsoft and Apple, include Root CA certificates in their Trusted Root Certification Authority certificate store that have met their security requirements. Some applications use the OS certificate stores, while others (e.g. Firefox and Oracle) have separate certificate stores.

Although OS and application vendors have a list of Trust Root CA certificates, additional Root CA certificates may be added to the Trusted Root Certification Authority certificate store by Administrators and end users with required security permissions. If a Root CA is compromised or no longer trusted, security patches shall be applied to remove them from the Trusted Root Certification Authority certificate store or they shall be removed manually.

**Real world Examples for the dangers of accepting a root certificates**

A recent Google warning over fake SSL certificates demonstrates, how one ‘rogue’ CA issuing unreliable certificates can cause havoc. Unfortunately, Certification Authorities can (and have been known to) issue fake certificates.

Trusting a malicious root is one of those nuclear-level “game over” scenarios. In fact, Chromium (the open-source project Google’s Chrome is based on) acknowledges that if an attacker can install a Root CA certificate onto your device, there is nothing the browser can do to protect you.

Current versions of Windows and OSX browsers provide notifications when a Root CA is not trusted. Trusting the Root CA requires manual installation of the Root CA certificate. For Windows, this takes quite a few steps, including viewing the Root CA certificate in the certificate chain and using the certificate import wizard.

On iOS, it is a different story. In Safari, just clicking a button on a webpage can prompt a system dialog to install a custom “profile” which can include Root CA certificates. An example of a profile that contains Root CA certificates is Comcast’s Xfinity Wifi. While iOS 10.3 and later does not automatically trust the certificate for SSL, earlier versions do trust the certificate.

User added Root CA certificates could be used maliciously. The new certificates could be used in a man-in-the-middle attack. This has always been a known vulnerability, but has not been of major concern because while feasible, it’s impractical in most attack scenarios.

**Managing trust and root certificates.**

Because of the growing variety of certificates in use today and the growing number of certificate issues, some organizations may want to manage certificate trust and prevent users in the domain from configuring their own set of trusted root certificates. In addition, some organizations may want to identify and distribute specific trusted root CA certificates to enable business scenarios where additional trust relationships are needed.

Adding a Root CA certificate to the Trusted Root Certification Authority certificate store can be quite labor intensive. The scale for IoT applications that could connect from a variety of issuing CA’s could be overwhelming for an organization. As indicated previously with the possibility of many IoT application developers using certificates from the same vendors, one root CA certificate for all could be dangerous, where as managing intermediate certificates for individual application vendors would be overwhelming.

**Using App-ID Registry Function as a trusted root for the Service provider infrastructure**

When an App-ID is registered, the metadata registered with the App-ID can contain information regarding the PKI structure, including:

* Issuing CA
* Intermediate CA certificate, if used
* Root certificate and its validity
* Location of the CRL ( certificate revocation list )

The SP infrastructure can query a connecting App-ID /AE-ID identity from the App-ID Registry Function to collect the PKI metadata, to verify the authentication procedures of oneM2M ( MAF) .

* The SP infrastructure can verify that the Root CA certificate is valid and can be trusted (to what level)
* That the issuing CA is valid and is still supported by the root CA.
* If an intermediate certificate is required to authenticate the IoT application certificate.
* Verify with the CRL that the certificate is not revoked.

The App-ID Registry Function can act as the root of trust for the oneM2M infrastructure, enabling the SP infrastructure to authenticate registered IoT applications through an automated process.

## 6.6 Use Case 6: Revoke App-ID or AE-ID

### 6.6.1 Description

The App-ID Registry may decide to:

* **Revoke an App-ID:** in this case a registered AE that has the revoked App-ID value will be denied a service or deregistered by the registrar CSE at the request of the App-ID Registry. A typical example that motivates revoking an App-ID is: when an application (corresponding to the revoked App-ID) is declared permanently or temporary unauthorized. For example when a device is detected to be misbehaving (e.g. sending unexpected data), an administrator may decide to put it into quarantine until it is fully checked. Alternatively the administrator can decide to declare an application software to be permanently invalid through revoking the corresponding App-ID. In this case, typically, a software upgrade will be triggered and a new version of the software, with a new App-ID value, will be installed in the devices. The revocation of an App-ID means all registered AEs with that App-ID value shall be denied access until further notice by the App-ID Registry.
* **Revoke an AE-ID:** in this case only a particular AE-ID will be denied service. The App-ID Registry typically informs the registrar CSE about the need to declare an AE registration as revoked.

## 6.7 Actors

The entities involved in this use case are shown in the Figure 5.9-1 and described as follows:

**M2M Service Infrastructure:** It represents a Service Provider infrastructure, including the physical equipment (e.g. a set of physical servers) that provides management of data and coordination capabilities for the M2M Service Provider and communicates with M2M Devices

NOTE: An M2M Service Infrastructure may communicate with other M2M Service Infrastructures. An M2M Service Infrastructure contains a CSE. It can also contain M2M applications.

**M2M Service Provider:** entity (e.g. a company) that provides M2M Common Services to a M2M Application Service Provider or to the User

**M2M Application:** It represents an IoT application that is responsible for accumulating data, transferring it to the SP and performing actions instructed by the SP .

**M2M App-ID Registration Authority (ARA):** legal entity that manages/administers the App-ID database used to issue unique global identifiers consistent with oneM2M specifications.

## 6.8 Pre-conditions

The following is a list of pre-conditions:

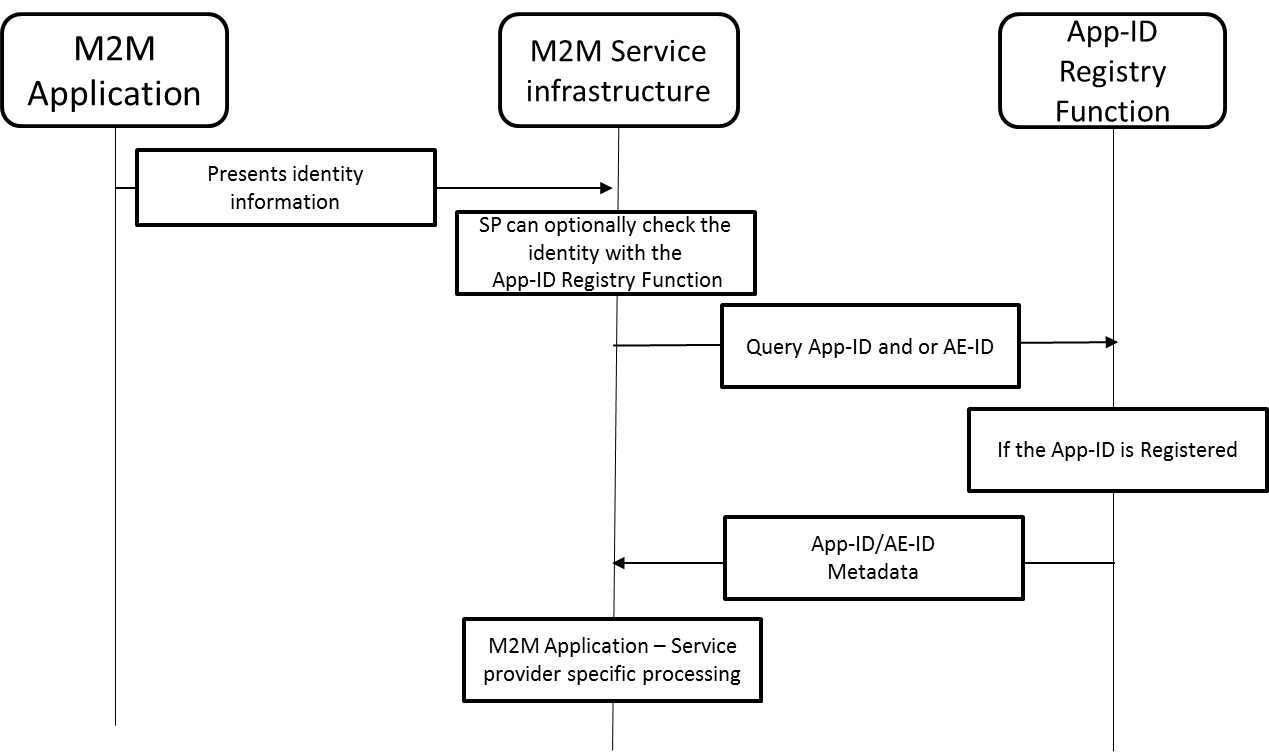
* The M2M application is registered in the App-ID Registry Function and has the metadata profile for its characteristics.
* The M2M Service Infrastructure is able to connect with the App-ID Registry Function to query the metadata for the M2M application identity.
* The App-ID Registry Function has metadata corresponding to the M2M application identity.
* The M2M Service Infrastructure is able to act upon the metadata returned from the App-ID Registry Function to process the M2M application
* The registration of an App-ID identity with the M2M Service infrastructure based on the following list of pre-conditions:
  + Variant 1: The App-ID Registry allowed registration of AEs (eventually more than one) which provided the necessary credentials to register with a specific App-ID value
  + Variant 2: The App-ID Registry allowed registration of an AE instance which provided the necessary credentials to register with a specific App-ID value and a specific AE-ID value

## 6.9 Normal Flow

Procedure to verify M2M application identity with App-ID Registry Function is:

1. The M2M application presents identity information to the M2M Service Infrastructure.
2. The M2M Service Infrastructure, if configured to verify the M2M application identity, will query the presented M2M applications App-ID and/or AE-ID to check if registered with the App-ID Registry Function.
3. If the M2M application identity is registered with the App-ID Registry Function, the registry will supply the metadata to the SP.
4. The M2M Service infrastructure reacts based on the metadata to process the associated M2M application according to the M2M service provider preferences, for example :

* Allow
* Disallow
* Allow with restrictions
* Trigger app registration
* Trigger app digital certificate verification
* Trigger comparison of a user’s privacy preferences and an applications privacy policies and new consent
* …… etc.



**Figure 6.9: Flow for query of App-ID and / or AE-ID metadata**

## 6.10 Revocation - Normal Flow

Procedure to revoke App-ID

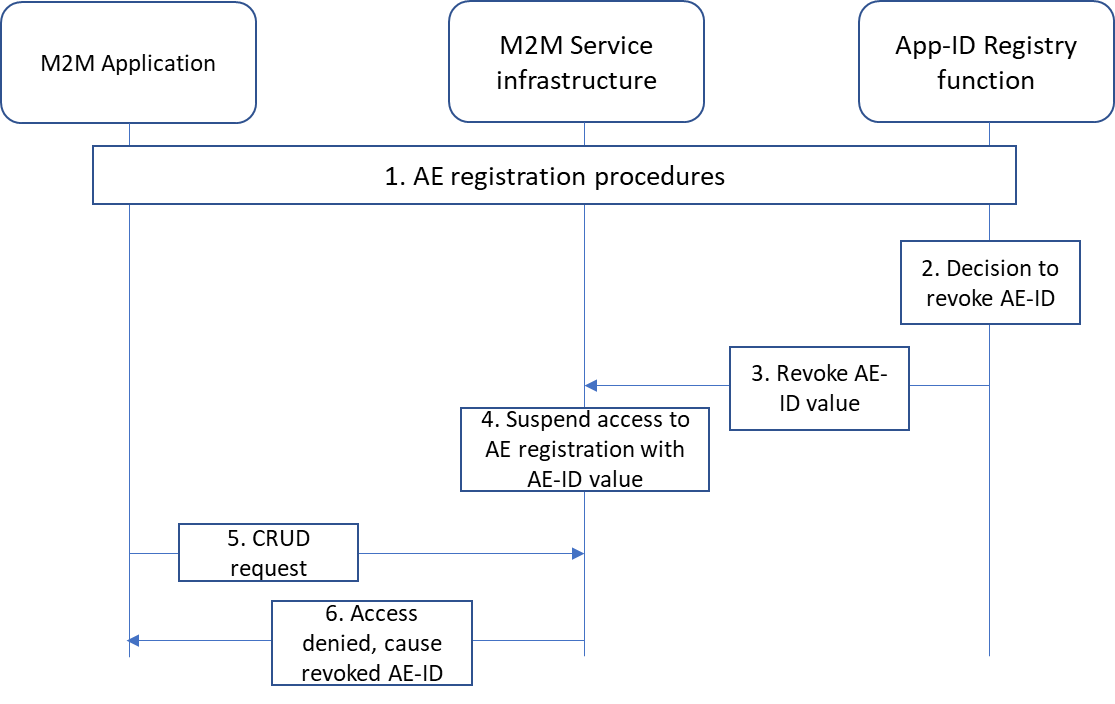
1. One or more AEs register to CSEs of M2M service infrastructure. App-ID Registry provides necessary authorisations as part of returned metadata
2. A specific event triggers a decision by the App-ID Registry to revoke an App-ID value, therefore suspending access to all AEs sharing that App-ID
3. App-ID Registry informs M2M Service infrastructure about the decision to Revoke and App-ID
4. M2M Service infrastructure may declare all AEs with the revoked App-ID value as suspended
5. and 6. All requests from AEs with the revoked App-ID are denied service



**Figure 6.10-1: Flow for revocation of App-ID**

Procedure to revoke AE-ID

1. An AE register to CSEs of M2M service infrastructure. App-ID Registry provides necessary authorisations as part of returned metadata
2. A specific event triggers a decision by the App-ID Registry to revoke AE-ID, therefore suspending access to AE
3. App-ID Registry informs M2M Service infrastructure about the decision to Revoke an AE-ID
4. M2M Service infrastructure may declare an AE registration with the revoked AE-ID value as suspended
5. and 6. All requests from AE with the revoked AE-ID are denied service



**Figure 6.10-2: Flow for revocation of App-ID**

## 6.11 Potential requirements

1. The M2M application shall be able to present the identity information with the M2M Service infrastructure to determine the App-ID and or AE-ID.
2. The M2M Service infrastructure shall be able to check the App-ID / AE-ID validity with the App-ID Registry Function.
3. The App-ID Registry Function shall allow registration of App-ID together with metadata.
4. The App-ID Registry Function shall support a metadata profile to characterize the M2M application .
5. The App-ID Registry Function shall provide a M2M Service infrastructure with the metadata for a registered App-ID.
6. The M2M service infrastructure shall be able to process the connected M2M application according to the provided metadata.
7. App-ID Registry shall keep track of registrar CSEs for all registered AEs (when App-ID Registry is used)
8. When requested by management action, the App-ID Registry shall request the revocation of App-ID or AE-ID.
9. A registrar CSE shall be capable of temporary or permanently suspend access to AEs with a revoked App-ID value
10. A registrar CSE shall be capable of temporary or permanently suspend access to an AE with a revoked AE-ID

# 7 Candidate Architecture

## 7.1 Candidate Architecture for App-ID Registry Function

The Candidate Architecture for the App-ID Registry Function to provide Metadata with the M2M Service infrastructure is as follows:

**OPTION #1** – **Standalone App-ID Registry Function (ARF)**

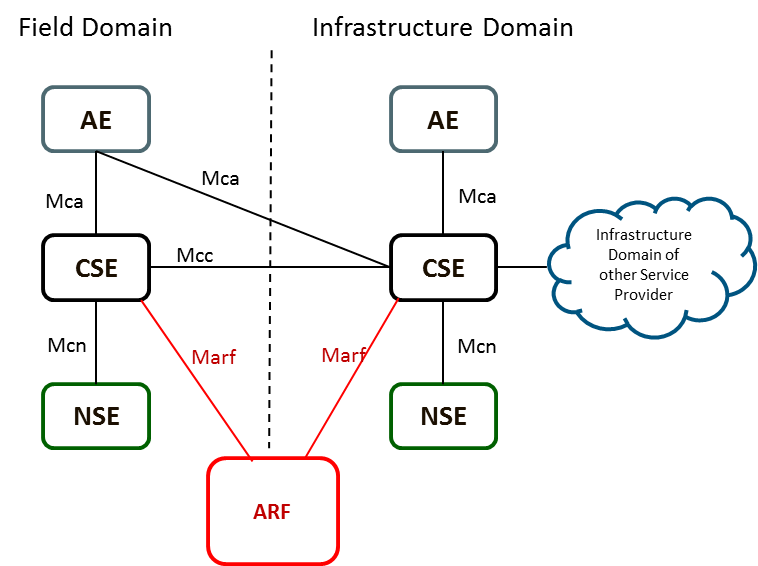
* The App-ID Registry Function is defined as a new entity of the oneM2M architecture.
* Defining a new interface for the CSE to query the App-ID Metadata.

**OPTION #2** – **App-ID Registry is part of existing oneM2M MEF interface specification**

* App-ID Registry Function that is part of an existing oneM2M architectural entity MEF.
* Update the current MEF interface to enable query of the App-ID Registry Function Metadata.

### 7.1.1 OPTION #1 – Standalone App-ID Registry function (ARF)

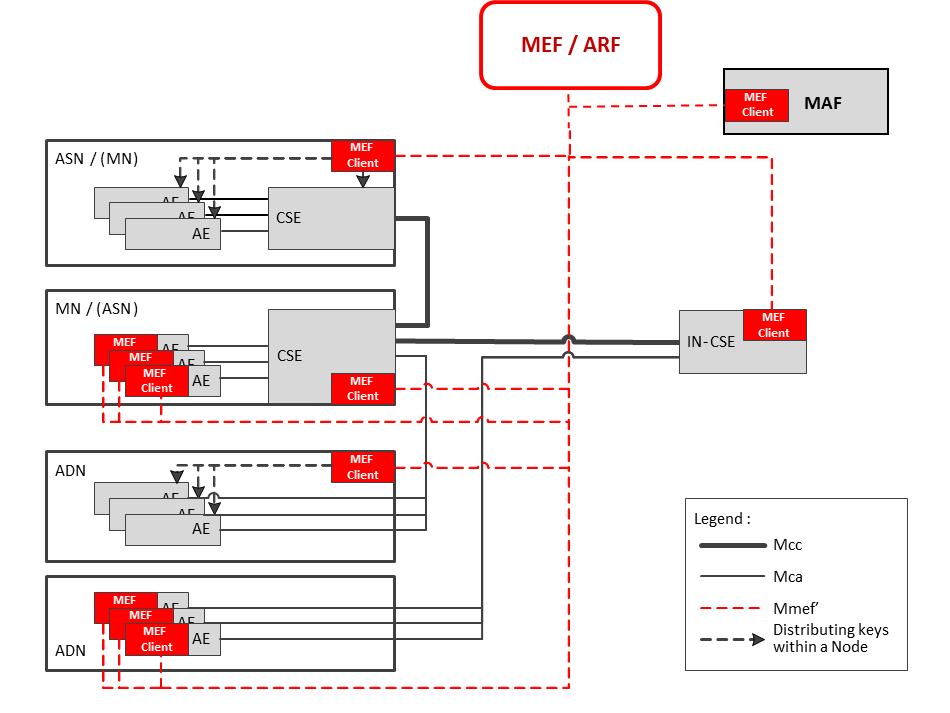
For the architecture of a standalone App-ID Registry Function, a new entity (ARF) and interface is defined as ‘Marf’.



**Figure 7.1.1: App-ID Registry Function new entity and interface**

### 7.1.2 OPTION #2 – App-ID Registry Function within MEF

For the App-ID Registry Function to be part of the existing oneM2M architecture, it is proposed that the ARF is an extended function of the MEF. The existing MEF entity and interface Mmef is extended and called Mmef’ (Mmef prime).



**Figure 7.1.2: App-ID Registry Function as part of the MEF entity and interface**

## 7.2 Architecture Proposal For Registration Enrolment using the App-ID Metadata

### 7.2.1 Functionality Requirements

The following is a list of basic requirements to be considered in design and analysis of Registration Enrolment using the App-ID Metadata solutions:

* Allow a Registree AE to enrol and be provisioned with a unique App-ID from an App-ID Registry
* Allow a Registrar CSE to perform an App-ID Registry lookup to check whether an App-ID it receives within a registration request is valid and has been assigned to the Registry.

### 7.2.2 Architecture Option #1

A candidate architecture is shown below in Figure 6.2.2-1. In this architecture, an independent App-ID Registry Function (ARF) and an ARF Client are defined. The ARF Client can be hosted by Registree and Registrar nodes. A Registree node can use it’s ARF Client to enrol to an ARF and be provisioned with App-ID(s) for the AE(s) the node hosts. A CSE hosted on a Registrar node can use it’s ARF Client to perform a lookup to the ARF to check whether a Registry AE’s App-ID is valid and in turn whether the Registrar CSE should grant or deny the Registree AE’s registration request. Using this lookup capability and the information returned by the ARF, the Registrar CSE can initiate the dynamic and remote provisioning of AE service subscription information (i.e. <serviceSubscribedAppRule> resources).



**Figure 7.2.2: Standalone App-ID Registry Function**

**Step 1:** ADN/ASN/MN’s MEF Client enrols to MEF as defined in TS-0003 and TS-0032 (e.g. <mefClientReg> and <symmKeyReg> resources are created).

**Step 2:** ADN/ASN/MN’s App-ID Registry Function (ARF) Client enrols to ARF such that ADN is provisioned with an App-ID.

**Step 3:** Security Association Establishment is performed

* **Step 3a:** AE initiates a Security Association Establishment handshake with its Registrar CSE. Depending on the type of security used this handshake may contain security related information. E.g. in the case of Provisioned Symmetric Key (PSK) based security, an identifier (KpsaID) is included.
* **Step 3b:** Registrar CSE formulates a Credential-ID based on the type of security that is used. E.g. in the case of PSK, the Credential-ID is formulated by pre-pending KpsaID with a value of “12-“. Based on the Credential-ID, the Registrar checks whether it has been provisioned with the Registree’s credential.
* **Step 3c:** If the Registrar CSE has not been provisioned with the Registree’s credential, the Registrar can initiate a lookup to the MEF to retrieve it based on the Registree’s Credential-ID.
* **Step 3d:** If the Registrar CSE has authorization to access the Registree’s credential, the MEF can respond with the credential.
* **Step 3e:** Using the credential, the Registrar can authenticate the Registree and complete the Security Association Establishment handshake.

**Step 4:** If the Security Association Establishment is successful, the AE can initiate a registration request to the Registrar CSE. Within this request, the Registrar provides its App-ID.

* **Step 4a:** If the Security Association Establishment is successful, the AE can initiate a registration request to the Registrar CSE. Within this request, the Registrar provides its App-ID
* **Step 4b:** Upon receiving the registration request from the AE, the Registrar CSE will check whether it has a matching <serviceSubscribedAppRule> resource for the AE. To find a match, the Registrar checks whether the Credential-ID and App-ID of the Registree AE match the *applicableCredIDs* and *allowedAppIDs* attributes of any <serviceSubscribedAppRule> resources it hosts. Also, if the registration request includes an AE-ID in the ***From*** request parameter, the Registrar CSE will also check whether this AE-ID matches the *allowedAEIDs* attribute of the <serviceSubscribedAppRule> resource. If a match is found, then the Registrar CSE will allow the AE to register.
* **Step 4c:** If a matching <serviceSubscribedAppRule> resource is not found for the AE, the Registrar can initiate a lookup to the ARF to check whether the AE has enrolled and been assigned an App-ID. This lookup can include information such as App-ID, Credential-ID and/or AE-ID.
* **Step 4d:** Using information provided in the request, the ARF can check whether the AE has enrolled and been assigned the App-ID included in the registration request. Based on the outcome of this check, the ARF can respond with an indication whether the AE is allowed to register. In addition, the ARF can provide additional information to the Registrar such as a list of allowed roles for the Registree, other allowed AE-IDs and Credential-IDs that are permitted to register using the same App-ID and/or a lifetime which the App-ID is valid for.
* **Step 4e:** Upon receiving a response from the ARF, the Registrar CSE can create or update a <serviceSubscribedAppRule> resource to store the information it received from the ARF such that it can be used to process the current registration request as well as future registration requests. For example, this information can be stored in the *applicableCredIDs, allowedAppIDs, allowedAEs, allowedRoleIDs* and the *expirationTime* attributes.
* **Step 4f:** Based on the outcome of the <serviceSubscribedAppRule> resource checks, the Registrar CSE either allows or denies the registration request.
* **Step 4g:** The Registrar CSE returns a registration response to the Registree AE that includes a status indication of whether the registration was successful or not.

#### 7.2.1.1 Nodes

This candidate architecture does not propose any new nodes to the oneM2M architecture. However, the following new architectural entities are proposed. The proposed entities can be hosted on the existing set of oneM2M defined nodes.

* App-ID Registry Function (ARF)
* ARF Client

#### 7.2.1.2 Reference Points

This candidate architecture proposes to define a new reference point between an ARF and an ARF Client (Marf).

### 7.2.3 Architecture Option #2

Another candidate architecture is shown below in Figure 6.2.3-1. In this candidate architecture App-ID Registry functionality is integrated into the MEF. A Registree node can use it’s MEF Client to enrol to a MEF and be provisioned with App-ID(s) for the AE(s) the node hosts. A CSE hosted on a Registrar node can use it’s MEF Client to perform a lookup to the App-ID Registry functionality integrated within the MEF to check whether a Registree AE’s App-ID is valid and in turn whether the Registrar CSE should grant or deny the Registree AE’s registration request. Using this lookup capability and the information returned by the MEF, the Registrar CSE can initiate the dynamic and remote provisioning of AE service subscription information (i.e. <serviceSubscribedAppRule> resources).



**Figure 7.2.3: App-ID Registry Function Integrated into MEF**

**Step 1:** Same as Step 1 in clause 6.2.2.

**Step 2:** Same as Step 2 in clause 6.2.2 except MEF Client is used by Registree node to enrol to ARF functionality integrated within MEF.

**Step 3:** Same as Step 3 in clause 6.2.2.

**Step 4:** Same as Step 4 in clause 6.2.2 except MEF Client is by Registrar node to perform lookup to ARF functionality integrated within MEF.

#### 7.2.3.1 Nodes

This candidate architecture does not propose any new nodes nor any new functions to the oneM2M architecture. The App-ID Registry functionality is instead integrated into the existing MEF.

#### 7.2.3.2 Reference Points

This candidate architecture does not propose any new reference points. Instead App-ID Registry functionality is added to the existing Mmef reference point.

7.3 Architecture Proposal For Management of Application Profiles

7.3.1 Functionality Requirements

The following is a list of basic requirements to be considered when specifying management of Application Profiles, and a list of design assumptions used for the proposed architecture solution.

An Application Profile refers to a set of information which characterizes an application in a compact and meaningful way for any stakeholder of an M2M system.

Requirements:

* In the M2M infrastructure a function is required which enables manufacturers and application software developers to register identifiers which characterize their products in a way meaningful to M2M network operators, M2M Service Providers, M2M Application Services Providers and M2M service subscribers:
  + a M2M Network Operator (i.e. operator of the underlying Wide Area Network, e.g. a mobile network) needs to know the characteristics (volume, distribution, message size) of the traffic created by an application
  + a M2M Service Provider (i.e. operator of the IN-CSE and associated infrastructure) needs to know the resource types created by an application, the expected amount of data volume created, the frequency and distribution of transactions triggered. The IN-CSE shall be enabled to authenticate any identifiers, including security credential identifiers, App-IDs, to authenticate and assign AE-IDs, and to associate identifiers with each other such as device, node, entity and application identifiers. It also shall check if operations requested by an AE are permitted in terms of the application user’s service subscription.
  + M2M Application Services Providers (i.e. the provider of value added services beyond plain Service Layer functions, typically the stakeholder who maintains a service agreement with both the M2M service subscribers and the M2M Service Provider). Examples include e.g. service providers of smart home services, Intelligent Transport Services (ITS), or eHealth services.
  + M2M service subscribers (i.e. individual end users who own M2M field devices or an organization which operates the M2M field system on behalf of the actual owner) may need to be enabled to discover other applications in order to interoperate and create added value
* An M2M system cannot rely on just a single central App-ID Registry Function. The architecture shall allow for multiple App-ID registries to coexist in the system preferably without need for communication between them.

Design assumptions:

* The App-ID Registry Function represents an infrastructure function which e.g. like the M2M Enrolment Function (MEF) may be operated by third party (i.e. a special M2M Application Services Provider) which configures information related to applications (Application Profiles) on the IN-CSE. This includes resources accessible via Mca and Mcc reference points such as <*m2mServiceSubscriptionProfile*>, <*serviceSubscribedNode*> <*serviceSubscribedAppRule*> and possibly in addition data which should not be exposed over Mca and Mcc. This implies that an App-ID Registry Function could include the functionality of an IN-AE for communication with the IN-CSE.
* The architecture solution considered aims to avoid direct communication between field entities and the App-ID Registry Function, in order to not burden field devices to support another client and interface for communication with the App-ID Registry Function. The solution described here would use the MEF as a proxy for any required communication between field entities and the App-ID Registry Function.
* For assignment and registration of App-IDs it is assumed that the procedures as defined today are continued to be used: the manufacturer / application developer proposes App-ID and the App-ID Registry Function validates uniqueness. The registered App-IDs is provisioned to the field device before it becomes operational in the M2M system.

### 7.3.2 Architecture

The proposed architecture solution is illustrated in Figure 6.3.2-1. The figure shows the functions and entities potentially having a role when considering the App-ID Registry Function. Solid lines define communication paths. These are referenced with a number in the figure and described as follows. Note that a practical implementation does not need to support all communication links shown in Figure 6.3.2-1. Note also that the numbering of these communication links is arbitrary and does not imply any sequence of steps to be executed in a specific order. Many certifications could be requested by the manufacturer or application developer/provider.

1) The manufacturer creates a digital certificate for his device, or an application software certificate, issuing a Certificate Signing Request (CSR) to a Certification Authority (CA). The certificate is initially installed on the field entity at manufacturing stage or software download/installation and used for Security Association Establishment with a MEF. The MEF-FQDN (i.e. the MEF ID) is also preconfigured on the device SW module.

2) Communication path for CSRs issued by the operational field node and communicated to the CA via the MEF (acting as PKI Registration Authority).

3) Communication path for CSRs issued by a Compliance Certification Body (CCB). In the proposed architecture, the CCB is assumed to issue a digital certificate upon having passed certification test successfully. This certificate may also be installed on the field entity and used as electronic proof that the device or software product is oneM2M certified. It includes all relevant identifiers, App-ID (which is assumed to be specific manufacturer/developer and allowing its identification), identification of the device (if applicable) and identification of the software instance (if applicable). Details of these identifiers are to be defined.

4) Communication path for CSRs issued by a Device Management (DM) Server. This function may be needed to ensure secure communication between a DM server and a DM client on the field node using OMA or BBF defined protocols (as defined in TS-0005, TS-0006 and TS-0022). This interface is out of scope of this document.

5) This refers to the communication path between manufacturer/developer and CCB to obtain the certificate described at no. 3) above. Details of this interface are out of scope of the present discussion.

6) The manufacturer/developer obtains the registered App-ID on this communication link from the App-ID Registry Function. After the manufacturer/developer has obtained a CCB-issued certificate which includes this registered App-ID, it returns this certificate to the App-ID Registry Function as proof of successful compliance registration. This information is stored as metadata in the Registry.

7) This refers to delivery of the device/software to the end customer (if the AE is implemented as Software module, it could be downloaded e.g. from a web store).

8) The manufacturer needs to agree with the operator of the MEF (provisioned to the device or SW module) on an applicable Security Association Establishment Framework (SAEF). Assumption in the following description is that certificate-based SAEF is used.

9) When the application is started for the first time, the MEF client contacts the MEF to receive security credentials and configuration data needed for registration to its registrar.

10) The MEF may trigger the field device to perform a DM procedure.

11) The DM server can configure the MEF to employ a specific authentication profile when communicating with a MEF client.

12) The DM server can be triggered by the IN-CSE to perform DM procedures with DM client on the field device, e.g. to configure CMDH policies.

13) If required, the MEF client on the field device could send information over communication path 9) which triggers a transaction between the MEF and the App-ID Registry Function. This feature would represent an extension to the currently given functionality on the MEF.

14) The App-ID Registry Function can send requests to the IN-CSE and receive responses. The App-ID would behave like an IN-AE on this communication path, i.e. Mca procedures would be applicable.

15) If the authentication profile applicable to the field device for registration indicates use of MAF-based SAEF, the field device establishes symmetric key credentials with a MAF. If certificate-based authentication between the field device and its registrar is configured to be used, the MAF is not involved.

16) When MAF-based SAEF is used, the registrar needs to retrieve the symmetric key credentials from the MAF, when the registree contacts the registrar.

17) When symmetric key credentials are assigned by the MEF for end-to-end security frameworks, the respective end-node needs to retrieve these credentials from the MEF (end-to-end security with symmetric key credentials could also be done with MAF procedures).

18) The field node may contact the IN-CSE either directly or via a Middle Node CSE. In the above descriptions it is assumed that a Middle Node is not present. The Middle node needs to register first to its own registrar (IN-CSE Figure 6.3.2-1, before the AE registers to the MN-CSE.

19) This communication path could be used to exchange information between App-ID Registry Function and CCB, for instance to define sets of testable functional features associated with Application Profiles for which compliance is certified by the CCB.



Figure 7.3.2: Nodes and communication interfaces of the proposed architecture

Summary of new features supported by the proposed solution:

* Validation if application has been certified by CCB by using digital certificates. This could be done by the App-ID Registry Function using communication via MEF, or by the IN-CSE (at registration, additional functions to be defined)
* CCB certificates could include information on tested functionality, which converts into service layer functions permitted to be used. These functions represent the *Application Profile*.
* App-ID Registry Function could act as IN-AE(s) towards IN-CSEs.
* For end.to-end security frameworks, App-IDs could be revoked by revoking App-ID certificates. New App-IDs could be assigned by installing new App-ID certificates on the AE, using MEF procedures triggered by the App-ID Registry Function. App-ID certificates could use the same private/public key pair as the AE's certificate. Once the AE's Certificate establishes a security association with the Registrar, the AE could create "App-IDCert" resources under its <AE> resources on the Registrar in order to have App-ID’s associated with the AE. (This could be made flexible by allowing the AE to create the certificates there and then allow activating/deactivating an individual App-ID).

#### 7.3.2.1 Nodes

The nodes of the overall proposed architecture are shown in Figure 6.3.2-1. Except for the App-ID Registry Function and CCB, all nodes are already specified in oneM2M Release 2A. The functionality of MEF clients and the MEF would require extensions. Some new resource types (e.g. on IN-CSE for App-ID metadata) and/or extensions to existing resource types (e.g. <AE> resource type) would need to be defined.

#### 7.3.2.2 Reference Points

The new reference points to be considered for standardization are the communication links of the App-ID Registry, marked with numbers 6), 13), 14 and 19). 6), and 13) may not need to be standardized. The communication link 14) between App-ID Registry Function and IN-CSE could be implemented compliant with the Mca reference point.

## 7.4 Architecture Proposal For functionality X

### 7.4.1 Functionality X Requirements

The following is a list of basic requirements to be considered in design and analysis of functionality X solutions:

### 7.4.2 Architecture

The Candidate Architecture for functionality X follows:

#### 7.4.2.1 Nodes

For the architecture of functionality X there are Y architectural components:

#### 7.3.2.2 Reference Points

There are Z reference points in the functionality Y Architecture:

# 8 App-ID Registry Function Metadata

The following clause describes the App-ID Registry Function metadata.

## 8.1 App-ID Registry Function – Metadata related Requirements

The following is a list of basic requirements to be considered in design and analysis of the App-ID Registry Function:-

1, The App-ID Registry Function shall allow registration of App-ID together with application metadata.

2, The App-ID Registry Function shall support a metadata profile to characterize the M2M application.

3, The App-ID Registry Function shall provide upon request a M2M Service infrastructure with the application metadata corresponding to a registered App-ID.

## 8.2 App-ID Metadata

The App-ID Registry Function provides metadata for each registered App-ID and the corresponding AE-IDs. The metadata assigned to the App-ID will be used to address the needs of the use cases described in clause 5. The following table provides the candidate list of App-ID metadata attributes defined into sub clauses x.x.

|  |  |  |  |
| --- | --- | --- | --- |
| **Group** | **Attribute** | **Description of use** | **Address Use Case #** |
| Type | Vertical application type | The vertical application type is used to determine if the application type is valid for the specific vertical service. Example types are:-   * Home Automation * Healthcare * Automotive * Energy * Industrial | Use Case: 1: to verify the identity of the type of device for access control.  Use Case 3: to allocate an appropriate <serviceSubscribedAppRule> for the type of application |
| Type | Application Unique Type | The specific type of application within the vertical, for example for Home Automation. Example types are :-   * Light Switch * Thermostat * Camera | Use Case: 1: to verify the identity of the type of device for access control.  Use Case 3: to allocate an appropriate <serviceSubscribedAppRule> for the type of application |
| Capability | Application Data Resources | How resource intensive the application is for projecting data. This metadata is used to create system resources appropriate to support the application data. | Use Case 3: to allocate an appropriate <serviceSubscribedAppRule> for the type of application |
| Data Definition | The application data model | The application native data model definition. This can be a URL to an XML file describing the data model.  Used to verify the application data / resource is the same way as the registered data model for the application identity. | Use Case 1: To verify the identity of the application, comparing the presented resources from the device with the reference metadata data model.  Use Case 3: to allocate an appropriate <serviceSubscribedAppRule> for the type of application |
| Data Definition | Mapping to oneM2M base ontology | Mapping of the applications native data model to a specific oneM2M base ontology to provide syntactic interoperability. | Use Case 4: to map the native data model to the base ontology |
| Security | Application Authentication Type | Authentication Type :- RPK, PSK, PKI  This metadata is used to establish the authentication capabilities of the connecting application and verify if the authentication type presented by the connecting application is the same as the registered metadata. | Use Case 1: To verify the identity of the application, comparing the presented authentication type with the registered metadata. |
| Security | Application Issuing CA | The URL of the issuing CA when applications are using PKI authentication.  This metadata is used to verify that the public key URL presented by the connecting application is the same as the registered metadata. | Use Case 1: To verify the identity of the application, comparing the presented public Key location with the registered metadata. |
| Security | Root CA assurance level | The Root CA assurance level for the issuing CA  This metadata is used to provide an indication of the assurance level for the Root CA that has validated the issuing CA according to oneM2M App-ID Registry management authority’s Certification Practice Statements (CPS) and Certificate Policy (CP) | Use Case 5: To provide the infrastructure with assurance assessment for the Root CA conformance. The infrastructure can use this to assess the level of trust of the issuing CA. |
| Security | MAF Identity | Where MAF is used for the App-ID instance authentication, the Identity of the MAF.  Used to identify the MAF entity for a presented App-ID. | Use Case 1: to provide identity for the applications associated MAF. |
| Security | MEF Identity | Where MEF is used for the App-ID instance enrolment, the Identity of the MEF.  Used to identify the MEF entity for a presented App-ID. | Use Case 1: to provide identity for the applications associated MEF. |
| Test & Compliance | Test and compliance entity name | The name of the test and compliance organisation that have verified the registered application identity. Note that this could be one or more entities, i.e. oneM2M compliance and / or an industry specific compliance body, e.g. healthcare.  This metadata can be used to determine if a connecting application has the relevant test and compliance to support the service. | Use Case 2: Allow the infrastructure to determine test and compliance. |
| Test & Compliance | Test and compliance certificate | The compliance certification reference identity for this specific application | Use Case 2: Allow the infrastructure to determine test and compliance. |
| Revocation | App-ID Revocation | The specific reason code for why the App-ID was revoked.  Used to manage the full lifecycle of the App-ID to end of Life, and / or other reason for the App-ID to be revoked; for example if the App-ID software version has been compromised or faulty and is known to cause issues. This can be used to prevent these applications from obtaining service. | Use Case 6 (new contribution): to indicate that the App-ID is revoked with a reason code to allow the infrastructure to enforce accordingly. |
| Instance | AE-ID Range | The Range of AE-ID’s that are valid for this App-ID.  Provides a range of valid AE-ID’s that are valid and associated with this App-ID. | Use Case 1: To verify the unique instance identity of the application with the registered metadata. |
| Instance | Revoked AE-ID(s) | Revoked AE-ID identities. List of the AE-ID’s that have been revoked including the specific reason codes. Used to prevent an AE-ID from being authenticated.  Used to manage the full lifecycle of the individual AE-ID to end of Life, and / or other reason for the AE-ID to be revoked; for example if the AE-ID software has been compromised or faulty and is known to cause issues. This can be used to prevent a specific unique instance of an application identity from obtaining service. | Use Case 6 (new contribution): to indicate that the AE-ID is revoked with a reason code to allow the infrastructure to enforce accordingly. |

# 9 Available Options

## 9.1 Review of Existing Technology for use in the functionality

### 9.1.1 Review of Existing Technology N

#### 9.1.1.1 Introduction to Technology N

## 9.2 A Solution for providing Functionality A

### 9.2.1 General procedure for providing functionality A

# 10 Release X Function Rationale

## 10.1 Overview of Release X Features

The analysis in the preceding clauses were used to guide the specification of Release 2 features for App-ID Registry Function, which include the following:

## 10.2 Release X Function

### 10.2.1 X Function Overview

**Overview:**

### 10.2.2 X Functional Architecture

Functional architecture details for X

# 11 Conclusions and recommendations

The present document offers an overview of the use cases, requirements, architecture proposals and available solutions for an App-ID Registry Function.

Some of the contents have been normalized as Release X Technical Specification, as described in clause 8. Others may be used to facilitate future normative work resulting in oneM2M Technical Specifications.

# Annex A: Problem Statement for needing App-ID Registry Function

## A.1 Introduction

1. introduction

Use-case 1

Use-case 2

# History

|  |  |  |
| --- | --- | --- |
| **Publication history** | | |
| V.0.0.1 | 21-Sep-2017 | Initial Baseline for TR |
| V.0.0.2 | 11-Oct-2017 | New Baseline with agreed text for scope and use cases |
| V.0.0.3 | 4-Dec-2017 | Update with approved contributions at TP32 |
|  |  |  |
|  |  |  |