ETSI TS 103 780 V1.1.1 (2022-08)

SmartM2M;
SAREF: oneM2M usage guidelines

Reference
DTS/SmartM2M-103780

Keywords
interoperability, IoT, IoT platforms, oneM2M,
SAREF, semantic, smart lift

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document has the objective to provide guidelines for the usage of SAREF over oneM2M (also including the SDT interoperability) for vertical industry sectors.

The present document also provides a simple use case for guiding application developers to model physical devices in oneM2M and adding semantic annotations that will enable interoperability of devices that are modelled in oneM2M:

- Description of a physical device that is to be modelled in oneM2M.
- Description of methods that can be used to model the device using oneM2M resources and procedures.
- The semantic annotation of the devices using the oneM2M base ontology.
- The semantic queries that can be used to discover device capabilities and enable interoperability.
- The call flows for implementation of the use case with a focus on the semantic aspects.

2 References

2.1 Normative 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.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

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

Not Applicable.

2.2 Informative 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.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

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] ETSI TS 118 112: "oneM2M; Base Ontology (oneM2M TS-0012)".

[i.2] ETSI TS 118 130: "oneM2M; Ontology based Interworking (oneM2M TS-0030)".

[i.3] ETSI TS 118 104: "oneM2M; Service Layer Core Protocol (oneM2M TS-0004)".

[i.4] onem2m-jupyter-notebooks.

NOTE: Available at https://github.com/ankraft/onem2m-jupyter-notebooks.

[i.5] ETSI TS 118 123: "oneM2M; Home Appliances Information Model and Mapping (oneM2M TS-0023)".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AE	Application Entity
API	Application Programming Interface
CSE	Common Service Entity
HTTP	Hypertext Transfer Protocol
IoT	Internet Of Things
IPE	Interworking Proxy Element
JSON	JavaScript Object Notation
M2M	Machine to Machine
NoDN	Non-oneM2M Device Node
RDF	Resource Description Framework
SAREF	Smart Applications REFerence ontology
SDT	Smart Device Template
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
X-M2M-CTS	oneM2M HTTP protocol custom header for "Content Status"
X-M2M-RI	oneM2M HTTP protocol custom header for "Request Identifier"
X-M2M-RSC	oneM2M HTTP protocol custom header for "Response Status Code"
X-M2M-RVI	oneM2M HTTP protocol custom header for "Release Version Indicator"
XML	eXtensible Markup Language

4 Motivation

The assumption of many existing oneM2M applications is that they interact with other oneM2M applications through known resource structures. They either create the resources themselves or are configured to use specific resources. Information is typically stored in containers, sometimes as base64-encoded content instances, with the implicit assumption that applications have a-priori knowledge of the syntax and semantics of this information.

Depending on a-priori knowledge of the structures and data works well for small-scale vertical deployments of IoT devices. When the deployment evolves to include new devices, the existing applications change to reflect the new additions. However, in larger systems of IoT devices where the IoT devices may be a part of a legacy deployment or more than a single vertical solution, changes to all existing applications may become impractical. To enable growth and diversity of IoT devices in large heterogenous settings, applications need to be able to **discover** the structure and meaning of data from devices and how to use the services of the devices. In oneM2M Release 1 support for discovery of resources based on specific attribute values and the use of labels was defined. The agreement of a fixed set of labels (using a-priori knowledge) can be a viable solution for small deployments.

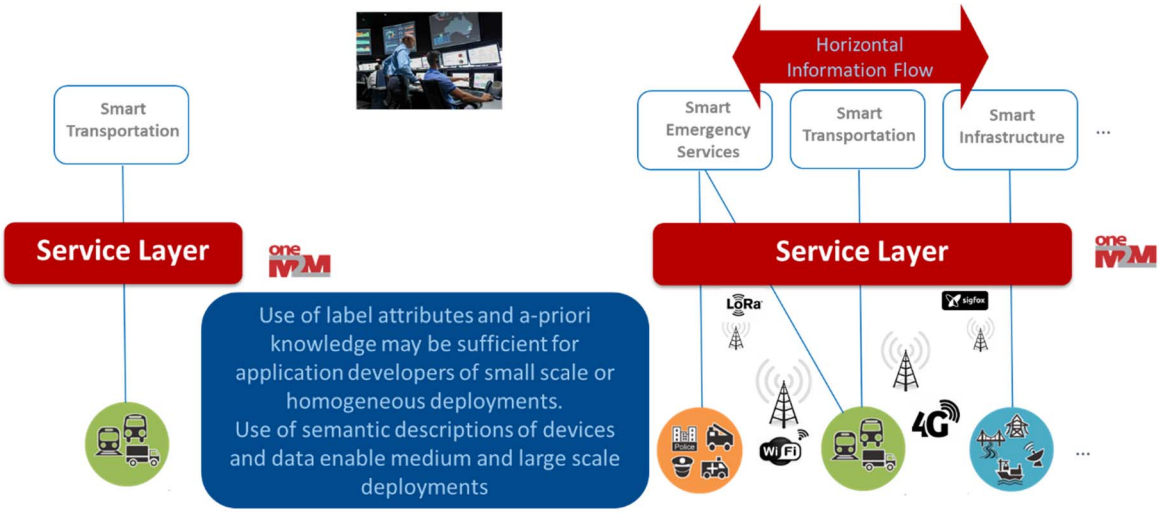

Figure 1: Semantic understanding of device and data in IoT deployments

For medium or large deployments of heterogeneous IoT devices a more expressive approach for describing and discovering IoT devices is provided by oneM2M. Each type of device in a heterogeneous deployment can model services and data in the oneM2M Service Layer using different structures and syntax of data. For example, temperature sensors may report measurements using different units such as Celsius, Fahrenheit and Kelvin. Additionally, those IoT devices may measure different aspects, such as indoor temperature, outdoor temperature, refrigerator temperature, etc. and the representation of the measurement may differ as well.

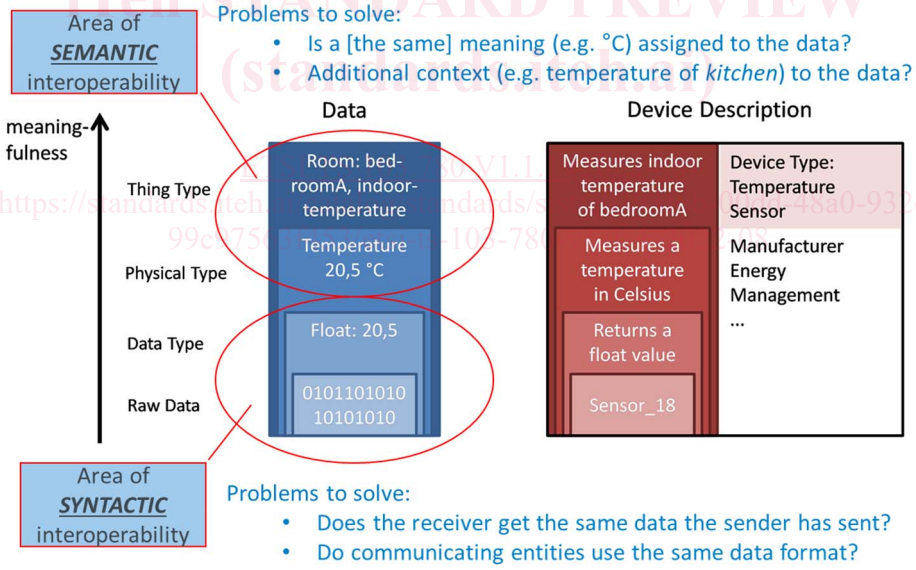

Figure 2: Meaningfulness of data from IoT devices

With semantic annotations in oneM2M, all the different aspects of IoT devices can be described using RDF triples, which is a standard semantic format. The vocabulary used for a semantic description can be defined according to an ontology such as SAREF. With semantic discovery, applications can describe precisely what information they need or can deal with. This is powered by specifying a semantic filter using the SPARQL query language. The SPARQL filter is matched against the respective semantic annotations of each resource within the discovery scope. This feature in oneM2M helps applications to properly handle the data from the IoT devices.

Besides differences in the data from an IoT device in oneM2M the information model of devices can be modelled in a variety of ways. As with most IoT platforms, oneM2M supports custom information models that are defined for a specific use case and work well in small scale or single vertical scenarios. Another method that oneM2M defines to model devices is based on the semantic description of a device that is mapped to a resource structure (see ETSI TS 118 130 [i.2]). A third approach to modelling devices in oneM2M is the use of Smart Device Templates (see ETSI TS 118 123 [i.5]).

With all these options available to model a device the ability to have a-priori knowledge of a device model becomes less likely as IoT deployments scale beyond small vertical use cases. The oneM2M Base Ontology addresses this and enables developers of these different models to make them interoperable if the appropriate semantic annotations are made and semantic filtering is used to discover the appropriate API for a model. The focus of the remainder of this developer guide is to demonstrate this process.

5 System Description

5.1 Use case

The example scenario describes a clothes-washing machine and an application to monitor and control the IoT enabled product. This clause will show three different oneM2M resource tree models of the clothes washing machine and the call flows to create those models. The logic and call flows necessary for a client application to control and monitor the status of the clothes-washing machine is also described. In the next clause the washing machine capabilities are described using the SAREF ontology so that the client application can discover the washing machines. Additionally, the oneM2M Base Ontology describes how to use the device and commands that these clothes washing machines offer so that the client application can control any of them without regard to which resource tree model represents them.

This simplified clothes-washing machine has enough features to demonstrate the difference between the different modelling approaches supported in oneM2M. The concepts shown here can be applied to a full featured clothes-washing machine or any other IoT enabled device for that matter. The features and capabilities that are modelled are:

- The washing machine has been produced by manufacturer **XYZ.**
- XYZ describes this type of washing machine as "`Very cool Washing Machine`".
- The model of the type of washing machine is **XYZ_Cool.**
- The state of the washing machine can take the values "WASHING" or "STOPPED" or "ERROR".
- The washing machine supports three commands: **ON, OFF, Toggle.**
- The washing machine is in **My_Bathroom.**

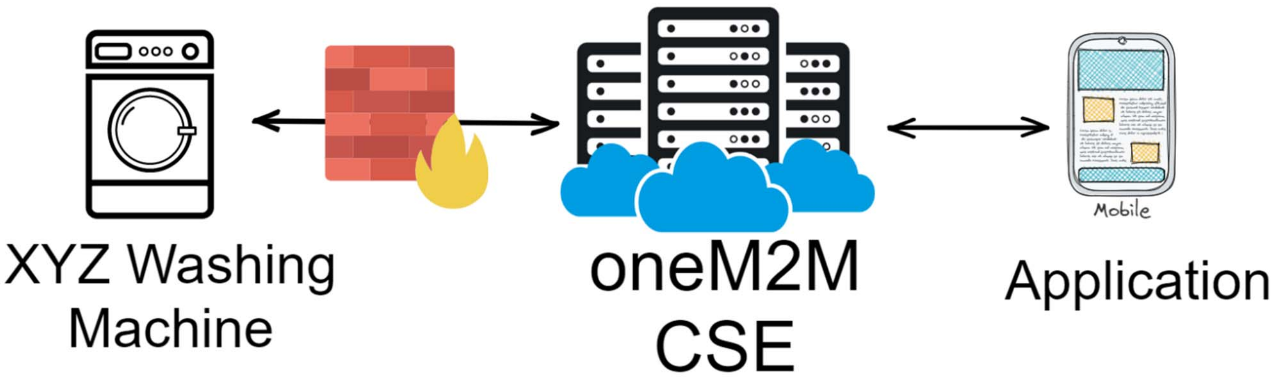

Figure 3: Functional Architecture for Smart Clothes Washing Machine

The clothes-washing machine is modelled as a non-oneM2M device (NoDN) for all three models. However, everything in this guide applies equally if these were modelled as native oneM2M devices. There is no difference in the model or the call flows for everything to the right of the Interworking Proxy Element (IPE) shown in the figure below. Figure 4 shows a generic set of oneM2M call flows for the clothes washing machine (and the IPE) and the client application communicating through the oneM2M CSE. The level of detail provided here applies to all the different modelling approaches for the clothes washing machine. Differences in the call flows that are dependent on the model used, shown in blue shading, are further detailed where the specific models are described.

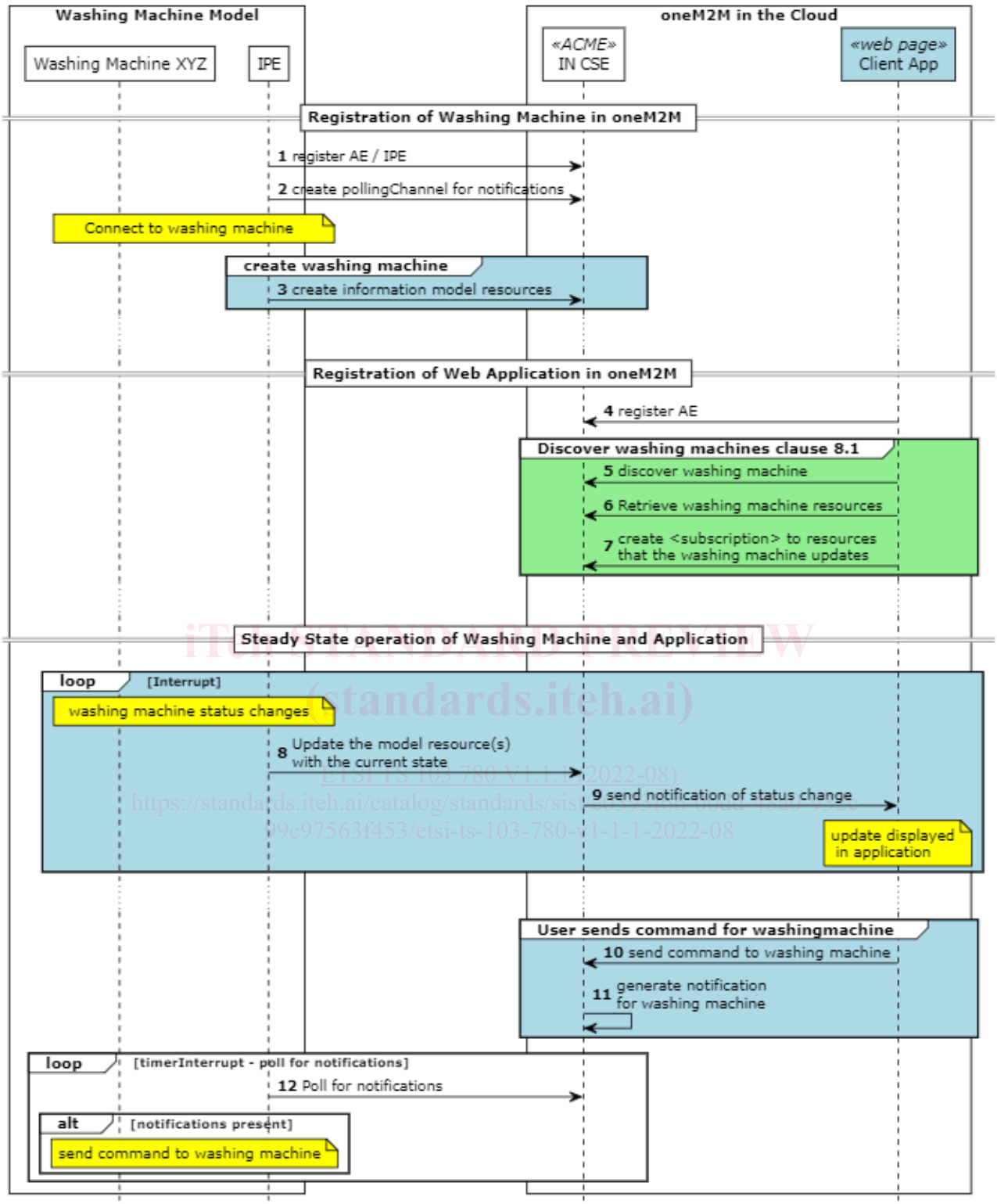

Figure 4: Generic oneM2M Call Flows

The messages shown in Figure 4 are further described here:

1. Register the AE/IPE: In oneM2M an IPE is a type of AE that is intended to communicate with NoDNs. The IPE is responsible for registering itself and creating the appropriate resources in a oneM2M CSE to model the NoDN as if it were a oneM2M device. The result is that a washing machine that is native oneM2M and a washing machine that is non-oneM2M can be modelled the same way and the client applications cannot tell the difference.

2. Create Polling Channel: A <pollingChannel> resource is used by applications or devices that are not reachable from the CSE that need to receive notification requests. This happens when, for example, the device is in a home with a firewall that prevents direct requests to the device from outside the local network in the home. (It is also appropriate for IoT devices that communicate using cellular networks.)

3. Create Information Model: The IPE creates all the resources needed to provide the status and enable control of the clothes washing machines. These messages (in almost all cases multiple resources are used) will be described with the details relevant to the specific model in later sub-clauses. This includes creating subscriptions to the resources that are used to enable the application to control the clothes washing machines.

4. Register Client application AE: Client applications are also modelled as <AE> resources and register in the oneM2M CSE.

5. Discover Washing Machine: An application designed to control the clothes washing machines produced by manufacturer XYZ will be able to discover them using a-priori knowledge of labels that are used to identify those washing machines. Later it has been shown how using the semantics capabilities of oneM2M and the SAREF ontology the same application can discover and control clothes washing machines from any manufacturer.

6. Retrieve clothes washing machine resources: The client application generally has a user interface to show the status and allow control of the clothes washing machine. The client application will retrieve the specific resources that it needs to provide that capability. The application may have more features than a given washing machine model supports or, similarly the clothes washing machine model may expose more features than the client application needs. This step will use SPARQL queries to dynamically determine what resources are needed by the client application.

7. Subscribe to resources: The client application is made aware of changes in the state of a clothes washing machine by receiving notifications of the changes. The client application first subscribes to the resources that contain information that it needs.

8. Update the model resources: When the state of the clothes washing machine changes, the change in state will be reflected in the oneM2M CSE.

9. Notification of state changes: When resources in the oneM2M CSE are created or updated the CSE will send notifications to applications that are subscribed to the resources. A client application that receives a notification can present this information to users or take some other actions.

10. Send commands to clothes washing machine: The client application exposes to a user features or capabilities of the clothes washing machine. The client application sends the appropriate oneM2M primitives, based on the model, to use those features or capabilities.

11. Generate notification for clothes washing machine: When the client application sends a oneM2M primitive to a resource that controls the clothes washing machine, a notification is generated (assuming notifications were created). In our scenario, since the clothes washing machine and the IPE are behind a firewall and therefore not reachable, the notification for the IPE are stored in the CSE and made available to the IPE via the long polling process.

12. Poll for notifications. Because the IPE cannot receive notifications directly, it shall use the long polling procedure to retrieve its notifications from the CSE. The IPE processes notifications by sending commands to the clothes washing machine using the API of the clothes washing machine.

5.2 Custom Model

Using oneM2M to represent devices allows for unlimited flexibility. A device model can be customized to support the needs of the manufacturer or system architecture. The resource tree structure shown here represents a custom model that has a single container for reading the status of the washing machine and a separate container to set or command the washing machine.

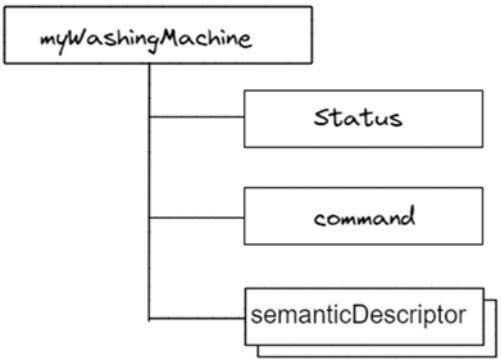

Figure 5: Custom washing machine model

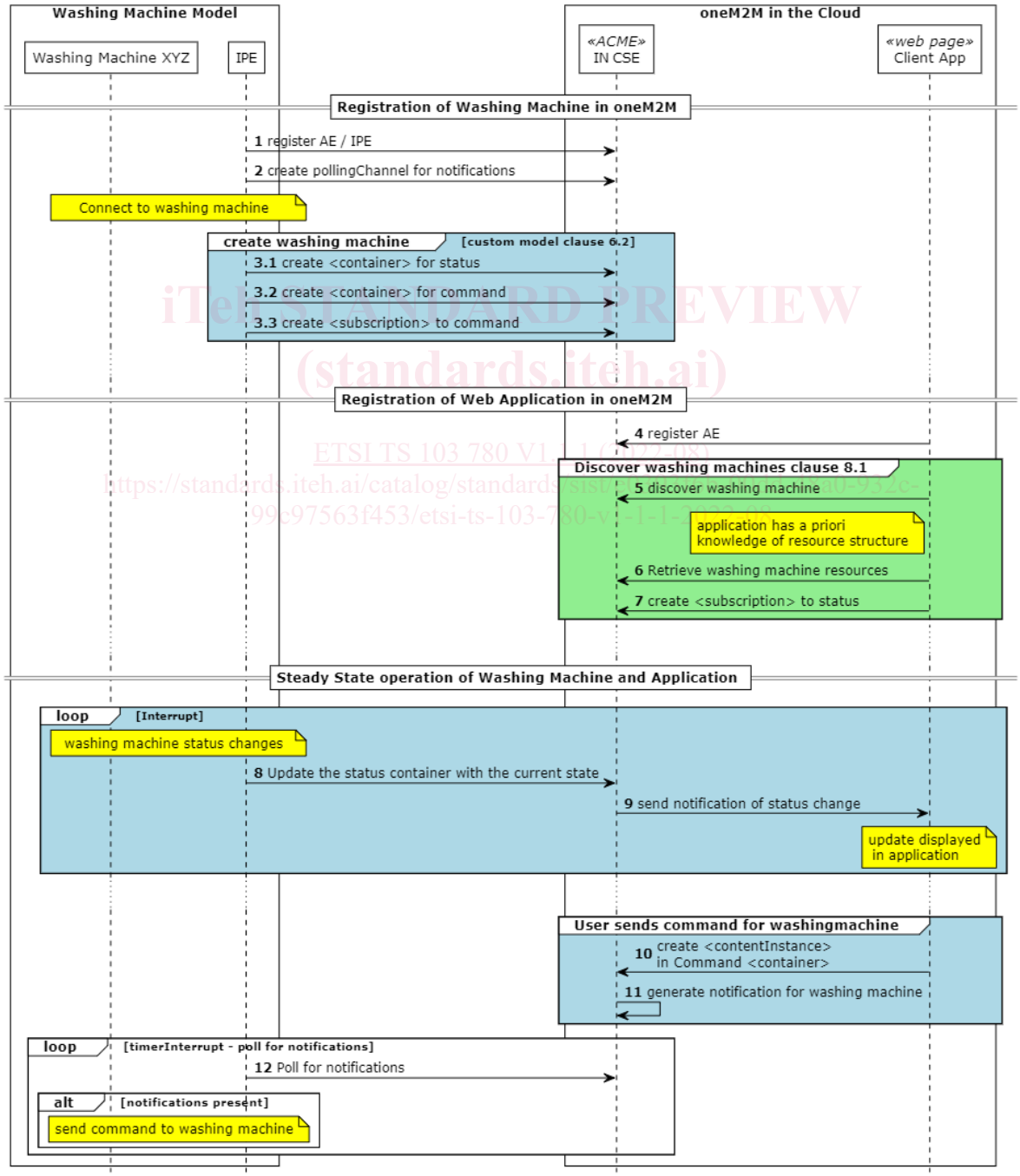

Figure 6: Custom Model oneM2M Call Flows

Only the messages highlighted in light blue are described here as the rest of the messages are the same as in the general call flow described in clause 5.1.

Create Information Model (Figure 6): The IPE creates all the resources needed to for the clothes washing machine that it knows how to model a priori. This IPE is developed with awareness of the clothes washing machine interface and the model that it is creating in the oneM2M CSE.

A <container> resource is created for the Status information of the clothes washing machine. The IPE creates <contentInstance> resources that have the following content when there are any changes in the status of the clothes washing machine:

```
{
    "WashingMachineStatus ": "WASHING", // Or "STOPPED", "ERROR"
}
```

A <container> resource is created for the command and control of the clothes washing machine. When the client application is setting the state of the device the following payload can be provided in a <contentInstance> resource:

```
{
    "state": "ON", // Or "OFF", "Toggle"
}
```

A <subscription> resource is created as a child of the command <container> resource by the IPE. This will cause a notification to be sent to the IPE when a new command is made by an application.

5.3 Semantic Modelling

A SAREF description of the washing machine is mapped to the resource structure shown in Figure 7 using the rules described in ETSI TS 118 130 [i.2] and ETSI TS 118 112 [i.1]. A complete derivation of this example is shown in ETSI TS 118 112 [i.1], clause B.1.3.3.

The description of our (simplified) washing machine using SAREF ontology is expanded upon here:

- The state of the washing machine is given by SAREF:state: **WashingMachineStatus** that can take the values "WASHING" or "STOPPED" or "ERROR".
- The washing machine has an actuating function: **StartStopFunction** which has three commands:
  - **ON_Command**
  - **OFF_Command**
  - **Toggle_Command**
- The washing machine has also a metering function: **MonitoringFunction** that sets the WashingMachineStatus.
- The washing machine is located at **My_Bathroom.**

Later it is shown that the description here has triples that are intended to help define the resource tree structure according to the rules described in ETSI TS 118 130 [i.2] and ETSI TS 118 112 [i.1]. However, when the description of the clothes-washing machine is put into a <semanticdescriptor> some are removed because they do not offer information useful for SPARQL queries.