|  |  |
| --- | --- |
| Input Contribution | |
| Meeting ID\* | MAS 34 |
| Title:\* | Semantic Reasoning Function |
| Source:\* | Xu Li, Convida, [li.xu@convidawireless.com](mailto:li.xu@convidawireless.com)  Chonggang Wang, Convida, [wang.chonggang@convidawireless.com](mailto:wang.chonggang@convidawireless.com) |
| Date:\* | 2018-03-03 |
| Input related to\* | TR-0033-Study on Enhanced Semantic Enablement v.0.8.1 |
| Intended purpose of  document:\* | Decision  Discussion  Information  Other <specify> |
| Impacted other TS/TR(s) | n/a |
| Decision requested or recommendation:\* | The content is to be included in clause 8.7 of TR-0033 v.0.8.1. |
| Template Version: January 2017 (Do not modify) | |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

# Introduction

This contribution presents the following contents related to semantic reasoning, which is to be supported in oneM2M R4:

1. The semantic reasoning definition and the general introduction (See clause 8.7.1).
2. A hospital facilities surveillance use case in the smart city scenario (See clause 8.7.2.1).
3. Technical analysis and needs for semantic reasoning in oneM2M. Two examples are presented in the context of the hospital facility surveillance use case in order to demonstrate why oneM2M system needs semantic reasoning feature and the unique characteristics/challenges for enabling semantic reasoning in oneM2M systems (See clause 8.7.2.2).
4. Based on the analysis in clause 8.7.2, a new Semantic Reasoning Function (SRF) is proposed and three key features of SRF are presented and each of those features are needed in order to support various application scenarios respectively (See clause 8.7.3):
   * The Feature-1 of SRF is to enable the semantic reasoning related data (referring to facts and reasoning rules) by making those data be discoverable, publishable/sharable across different users/entities in oneM2M system.
   * The Feature-2 of SRF is to leverage semantic reasoning as a “background support” to optimize other/existing semantic operations supported by oneM2M (such as semantic resource discovery, or semantic query, etc.).
   * The Feature-3 of SRF is to enable oneM2M users to directly interact with SRF by triggering individual semantic reasoning process (In other words, the semantic reasoning process is not necessarily coupled with other semantic operations as considered in Feature-2).

### -----------------------Start of change 1-------------------------------------------

## 8.7 Semantic reasoning

### 8.7.1 Semantic reasoning definition

RDF data/triple is the type of semantic metadata used in oneM2M system, which is in the form of 3-tuple representation, i.e., “subject-predict-object”. The RDF data is usefully written based on various ontologies, which define formal vocabularies (i.e., class) and their relationships for different application domains. Semantic reasoning can be leveraged to analyse RDF data and it is a mechanism to derive implicit facts that are not explicitly expressed in the existing knowledge/facts by leveraging a set of reasoning rules. There are several key concepts that are involved with semantic reasoning:

1. Existing facts – An existing fact is an already-known knowledge or an assertion.

2. Reasoning rules (RR) – A RR has an IF-THEN construct. When a given RR is applied over the existing facts, if the condition (i.e., the IF part) is true, then the conclusion (the THEN part) also holds.

3. Inferred facts – The output of a reasoning process, which is the reasoning result derived from the existing facts by applying RR.

As an example, a RDF triple, like “Flipper is-a Dolphin”, describes an existing fact about a dolphin named Flipper (i.e., Flipper is an instance of class Dolphin defined in an Animal ontology). In the meantime, the Animal ontology may declare another fact: every Dolphin is also a Mammal, which is a fact about a subclass relationship between two classes (i.e., Dolphin class and Mammal class). Giving a reasoning rule stating that “IF X is an instance of class Y and Y is a subclass of class Z, THEN X is also an instance of class Z (here X, Y and Z are variables and can be replaced with specific instance/class names)”, the following fact can be derived by applying this reasoning rule over the above two facts: Flipper is-a Mammal, which is an implicit knowledge inferred from the two existing facts.

To implement a semantic reasoning process for above example, a Semantic Reasoner (SR) is needed. Typically, a SR (also called reasoning engine, rules engine, or simply a reasoner) is a piece of software that is able to infer logical consequences from a set of facts using a set of reasoning rules.

### 8.7.2 Hospital facility surveillance use case

#### 8.7.2.1 Use case description

Figure 8.7.2-1 illustrates a hospital facility surveillance use case in the smart city scenario. In order to enforce surveillance and facility management purpose, a large hospital with many buildings has monitoring cameras installed in the rooms of those buildings. The hospital facilities are integrated into the city infrastructure (e.g., as an initiative for realizing smart city) such that external users (e.g., from city fire department, city health department, etc.) can also manage, query, operate and monitor facilities of the hospital. In each hospital building, rooms are used for different purposes. For example, some rooms (e.g., Room-232) are to store blood testing samples while other ones are to store medical oxygen cylinders. Due to the different usages of rooms, the hospital has defined several “Management Zones (MZ)” and each MZ likely comprises multiple rooms. Note that, the division of MZs is not necessarily based on geographical locations only, but also based on other room properties (e.g., room usage purpose). For example, MZ-1 includes all the rooms that store blood testing samples. Accordingly, those rooms will be more interested by city health department since the city health department may need to access the cameras deployed in the rooms belonging to MZ-1. Similarly, MZ-2 could consist of all the rooms, which store medical oxygen cylinders and accordingly will be more interested by city fire department. In addition, rooms in each MZ may be changed over time due to room rearrangement or re-allocation by the hospital facility team. For example, Room-232 may belong to MZ-2 when it starts to be used for storing medical oxygen cylinders, i.e., not storing blood test samples any more.

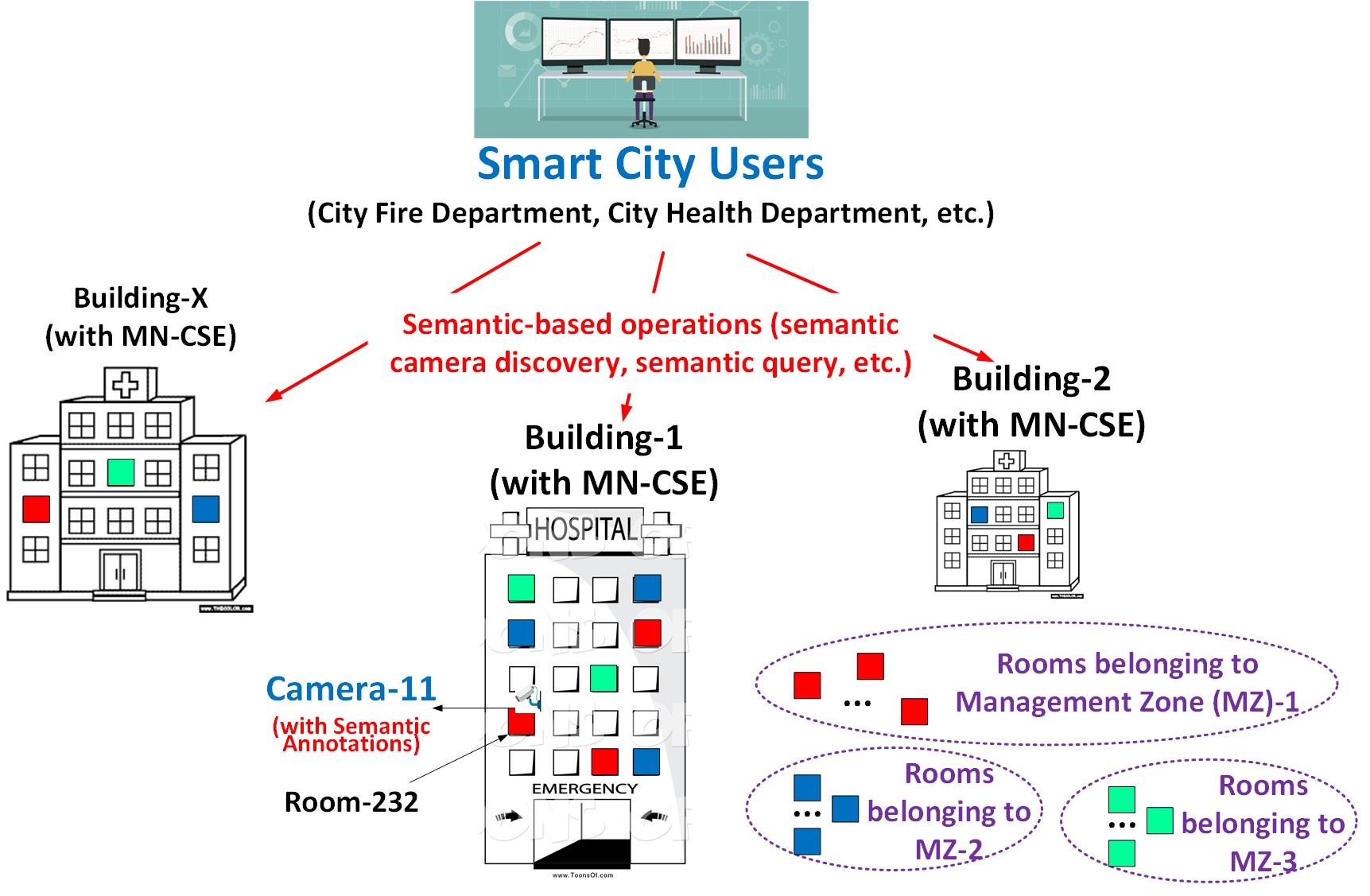


Figure 8.7.2-1: Hospital Facility Surveillance Use Case

#### 8.7.2.2 The needs for semantic reasoning in oneM2M

In the following, two examples, based on the hospital facility surveillance use case, are described to demonstrate why oneM2M system needs semantic reasoning feature. Unique challenges for enabling semantic reasoning in oneM2M systems are also discussed. Both examples mainly focus on Camera-11 as shown in Figure 8.7.2-1, which is deployed in the Room-232 of Building-1. The following assumptions are made:

* The hospital has adopted an oneM2M-based platform to manage its facilities including cameras.
* Each building hosts an MN-CSE.
* Each camera deployed inside a building registers to the corresponding MN-CSE of the building.
* Each camera is represented as a service layer resource at the MN-CSE. For example, Camera-11 deployed in Room-232 of Building-1 has a <Camera-11> resource representation hosted on the MN-CSE of Building-1, which is an <AE> resource.
* The <Camera-11> resource is annotated with some metadata by adding a <semanticDescriptor> resource as its child resource. In particular, the <semanticDescriptor> child resource stores two RDF triples (as existing facts):
  + RDF Triple #1 (i.e. Fact-1): Camera-11 is-a ontologyA:VideoCamera (where “VideoCamera” is a class defined by ontology A.
  + RFC Triple #2 (i.e. Fact-2): Camera-11 is-located-in Room-232-of-Building-1.

**Example 1**: Consider that a user needs to retrieve real-time images from all the rooms. In order to so, the user first needs to first perform semantic resource discovery to identify the cameras using the following SPARQL Statement-1:

SELECT ?device

WHERE {

?device is-a ontologyB:VideoRecorder

}

In reality, it is very likely that the semantic annotation of <Camera-11> and SPARQL Statement-1 may use different ontologies since they can be provided by different parties. For example, with respect to the semantic annotation of <Camera-11>, the ontology class “VideoCamera” used in Fact-1 is from Ontology A. In comparison, the ontology class “VideoRecorder” used in SPARQL Statement-1 is from another different Ontology B. Since semantic reasoning capability is missing, the system cannot figure out that ontologyA:VideoCamera is indeed as same as ontologyB:VideoRecorder. As a result, <Camera-11> resource cannot be identified as a desired resource during the semantic resource discovery process since the basic SPARQL processing is based on pattern matching (but in this example, the Fact-1 cannot match the pattern “?device is-a ontologyB:VideoRecorder” in the SPARQL Statement-1).

**Example 2**: A more complicated case is illustrated in this example, where the user just wants to retrieve real-time images from the rooms “belonging to a specific management zone (e.g., MZ-1)”. Then, the user also needs to first perform semantic resource discovery using the following SPARQL Statement-2:

SELECT ?device

WHERE {

?device is-a ontologyA:VideoCamera

?device monitors-room-in MZ-1

}

Similar to Example-1, due to the missing of semantic reasoning support, <Camera-11> resource cannot be identified as a desired resource either (at this time, Fact-1 matches the pattern “?device is-a ontologyA:VideoCamera” in the SPARQL Statement-2, but Fact-2 cannot match the pattern “?device monitors-room-in MZ-1”).

This example shows another reality that the semantic annotation of the devices may only include low-level metadata such as physical locations, and does not include high-level information such as the MZ-related fact in this example. However, the user may be just interested in rooms under a specific MZ (e.g., MZ-1), regardless of their physical locations. In the meantime, the user likely uses high-level ontology classes in their query statement (such as the predicate “monitors-room-in” in SPARQL Statement-2) while the semantic annotation of the devices may only use low-level ontology classes/relationships (such as the predicate “is-located-in” in Fact-2). Note that, such a phenomenon is very common due to the usage of “abstraction” technology in the sense that the query from upper-layer user is based on high-level concepts while lower-layer physical resources are annotated with low-level metadata. As an example, when a human user checks a file in a specific folder on her laptop, the operating system needs to locate the physical blocks of this file on the hard drive, which is fully transparent to the human user.

Example 2 also illustrates a critical semantic reasoning issue due to the lack of sufficient fact inputs for a reasoning process. For example, even if it is assumed that semantic reasoning is enabled and the following reasoning rule (i.e., RR-1) can be utilized:

* RR-1: IF X is-located-in Y && Y is-managed-under Z, THEN X monitors-room-in Z

Still, no inferred fact can be derived by applying RR-1 over Fact-2 through a semantic reasoning process. The reason is that Fact-2 can only match the “X is-located-in Y” part in RR-1 (i.e., to replace X with <Camera-11> and replace Y with “Room-232-of-Building-1”). However, in addition to Fact-1 and Fact-2, there is no further fact can be utilized to match “Y is-managed-under Z” part in RR-1 (i.e., there is no sufficient facts for using RR-1). In fact, the fact missing here is about hospital room allocation. The hospital room allocation records could be a set of RDF triples defining which rooms belong to which MZs, e.g., the following RDF triple describes that Room-232 of Building-1 belongs to MZ-1:

* Fact-3: Room-232-of-Building-1 is-managed-under MZ-1
* …..

Without Fact-3, semantic reasoning still cannot help in this example due to lack of sufficient facts as the inputs of reasoning process.

Overall, it is worth noting that in oneM2M system, it is highly possible that facts are often fragmented or distributed in different places. For example, the <semanticDescriptor> child resource of <Camera-11> may only store a limited amount of information about the camera itself and it does not make sense to stored RDF triples about hospital room allocation records directly in it (especially when room usage re-allocation happens from time to time). Therefore, there should be an efficient way to organize/integrate related “fact silos” in order to make all the facts ready when they are needed for a semantic reasoning process.

### 8.7.3 Semantic reasoning function (SRF)

This section describes the key features of Semantic Reasoning Function (SRF) in oneM2M system and the different type of functionalities that SRF shall support.



Figure 8.7.3-1: Key Features of Semantic Reasoning Function (SRF)

The key features of SRF is shown in Figure 8.7.3-1, In general, the SRF mainly includes the following features:

**Feature-1: Enabling semantic reasoning related data**

The major functionality of Feature-1 is to enable the semantic reasoning related data (referring to facts and reasoning rules) by making those data be discoverable, publishable/sharable across different entities in oneM2M system (which is illustrated as the dark yellow arrow in the Figure 8.7.3-1). The semantic reasoning related data can be a Fact Set (FS) and/or a Rule Set (RS). A FS refers to a set of facts. For example, each RDF triple can describe a fact, and accordingly a set of RDF triples stored in a <semanticDescriptor> resource is regarded as an FS. In general, a FS can be used as an input for a semantic reasoning process (i.e. an input FS) or it can be a set of inferred facts as the result of a semantic reasoning process (i.e. an inferred FS). A RS refers to a set of semantic reasoning rules.

To execute a specific semantic reasoning process A, the following two types of data inputs are essentially needed:

* An input FS (denoted as inputFS), and
* A RS.

The output of the semantic reasoning process A includes:

* An inferred FS (denoted as inferredFS), which is the semantic reasoning results of reasoning process A.

Note that, the inferredFS generated by a reasoning process A can further be used as an inputFS for another semantic reasoning process B in the future. Therefore, in the following descriptions, the general term FS will be used if applicable.

It is worth noting that, facts are not limited to semantic annotations of normal oneM2M resources (e.g., the RDF triples stored in <semanticDescriptor> resources). Facts can refer to any valuable information or knowledge that is made available in oneM2M system and can be accessed by others. For example, an ontology description stored in an oneM2M <ontology> resource can be a FS. Another case, a FS could also be an individual piece of information (such as the RDF triples describing hospital room allocation records as discussed in the previous use case in Figure 8.7.1.2-1), and such a FS is not either describing an ontology or as semantic annotation of another resource (i.e., the FS describing hospital room allocation records can individually exist and not necessarily be as the semantic annotations of other resources).

With regard to the RS, users have needs to design many customized (or user-defined) semantic reasoning rules for supporting various applications, since oneM2M system is designed to be a horizontal platform that enables applications across different domains. Accordingly, various user-defined RSs shall be made available in oneM2M system and can be accessed/shared by others. Note that, such user-defined semantic reasoning rules significantly improve the system flexibility since in many cases, the user-defined reasoning rules may be just used locally or temporarily (e.g., to define a new/temporary relationship between two classes in an ontology), which does not have to modify the ontology definition.

Overall, Feature-1 involves with enabling the publishing/discovering/sharing semantic reasoning related data (including both FSs and RSs) through appropriate oneM2M resources. The general flow of Feature-1 is that oneM2M users (as originator) can send requests to certain receiver CSEs in order to publish/discover/update/delete the FS/RS-related resources through the corresponding CRUD operations. Once the processing is done, the receiver CSE will send the response back to the originator.

**Feature-2: Optimizing other semantic operations with background semantic reasoning support**

Based on the two examples discussed in clause 8.7.2.2, the existing semantic operations supported in oneM2M system (e.g., semantic resource discovery and semantic query) may not yield desired results without semantic reasoning support. The major functionality of Feature-2 of SRF is to leverage semantic reasoning as a “background support” to optimize other semantic operations (which are illustrated by the pink arrows in the Figure 8.7.3-1). In this case, users trigger/initiate specific semantic operations (e.g., a semantic query). During the processing of this operation, semantic reasoning may be further triggered in the background, which is however fully transparent to the user. For example, a user may initiate a semantic query by submitting a SPARQL query to a SPARQL query engine. It is possible that the involved RDF triples (denoted as FS-1) cannot directly answer the SPARQL query. Accordingly, the SPARQL engine can further resort to a SR, which will conduct a semantic reasoning process. The SR shall determine and select the appropriate reasoning rule sets (as RS) and any additional FS if FS-1 (as inputFS) is insufficient, for instance, based on certain access rights. Finally, the semantic reasoning results in terms of inferredFS shall be delivered to the SPARQL engine, which can further be used to answer/match user’s SPARQL query statement.

By leveraging Feature-2, SRF can address the issue as illustrated in Example-1 in clause 8.7.2. For example, a reasoning rule (RR-2) can be defined as:

* RR-2: IF X is an instance of ontologyA:VideoCamera, THEN X is also an instance of ontologyB:VideoRecorder.

Here X is a variable and will be replaced by a specific instance (e.g., <Camera-11> in Example-1) during the reasoning process. When the SPARQL engine is processing the SPARQL Statement-1 defined in clause 8.7.2.2, it can further trigger a semantic reasoning process at the SR, which will apply the RR-2 (as RS) over the Fact-1 (as inputFS) as defined in clause 8.7.2.2. As a result, an inferredFS can be produced, which includes the following new fact:

* Inferred Fact-1: Camera-11 is-a ontologyB:VideoRecorder

The SPARQL engine now is able to use Inferred Fact-1 to match the pattern “?device is-a ontologyB:VideoRecorder” in the SPARQL Statement-1. As a result, with the help of SRF, <Camera-11> resource can now be identified as a desired resource during the semantic resource discovery.

The Feature-2 of SRF can also address the issue as illustrated in Example-2 in clause 8.7.2.2. For example, when the SPARQL engine processes SPARQL Statement-2 as defined in clause 8.7.2.2, it can further trigger a semantic reasoning process at the SR. In particular, the SR determines that RR-1 defined in clause 8.7.2.2 (as RS) should be utilized. In the meantime, the local policy of SR may be configured that in order to successfully apply the RR-1, the existing Fact-2 is not sufficient and additional Fact-3 should also be used as the input of the reasoning process (i.e., Fact-3 is a hospital room allocation record defining that Room-232 of Building-1 belongs to MZ-1). In this case, inputFS is further categorized into two parts: initial\_InputFS (i.e., Fact-2) and additional\_InputFS (i.e., Fact-3). As a result, by applying RR-1 over “the combined inputFS” (i.e., Fact-2 and Fact-3), an inferredFS can be produced, which includes the following new fact:

* Inferred Fact-2: Camera-11 monitors-room-in MZ-1

The SPARQL engine now is able to further use Inferred Fact-2 to match the query pattern “?device monitors-room-in MZ-1” in SPARQL Statement-2. As a result, <Camera-11> now can be successfully identified in the semantic resource discovery operation in Example-2 in clause 8.7.2.2.

Overall, the general flow of Feature-2 is that oneM2M users (as originator) can send requests to certain receiver CSEs for the desired semantic operations (such as semantic resource discovery, semantic query, etc.). During the request processing, the receiver CSE can further leverage reasoning capability. By using the reasoning result, the receiver CSE will further produce the final result for the semantic operation as requested by the originator (e.g., the semantic query result, or semantic discovery result) and then send the response back to the originator.

**Feature-3: Enabling individual semantic reasoning process**

In addition to the use cases as supported by Feature-2, semantic reasoning process may also be triggered individually by oneM2M users (which are illustrated by the blue arrows in the Figure 8.7.3-1). In other words, the semantic reasoning process is not necessarily coupled with other semantic operations as considered in Feature-2). With Feature-3, oneM2M users may directly interact with SRF by triggering semantic reasoning process. In order to do so, oneM2M user shall first identify the interested facts (as initial\_inputFS) as well as the desired reasoning rules (as RS) based on their application needs. When the inputFS and RS are identified, the oneM2M user shall send a request to SR for triggering a specific semantic reasoning process by specifying the reasoning inputs (i.e., the identified initial\_inputFS and RS). The SR will initiate a semantic reasoning process based on the inputs as indicated by the user. Similar to Feature-2, the SR may also determine what additional FS or RS needs to be leveraged if the inputs from the user are insufficient. Once the SR works out the semantic reasoning result, it will be returned back to the oneM2M user for its need. Typically, the following cases can be supported by Feature-3:

* Case-1: The oneM2M user may use SRF to conduct semantic reasoning over the low-level data in order to obtain high-level knowledge. For example, a health monitoring application (as an oneM2M user) can ask SRF to perform a semantic reasoning process over the real-time vital data (such as blood pressure, heart beat, etc.) collected from a specific patent A by using a heart-attack diagnosis/prediction reasoning rule. In this process, the heart-attack diagnosis/prediction reasoning rule is a user-defined rule, which can be highly customized based on patient A’s own health profile and his/her past heart-attack history. In this way, the health monitoring application does not have to deal with the low-level vital data (i.e., blood pressure, heart beat, etc.), and can get away from the determination of patient A’s heart-attack risk (since all the diagnosis/prediction business logics have already been defined in the reasoning rule used by SRF). As a result, the health monitoring application just needs to utilize the reasoning result (i.e., the patient A’s current heart-attack risk, which is a “ready-to-use or high-level” knowledge) and send an alarm to doctor or call 911 for an ambulance if needed.
* Case-2: The oneM2M user may use SRF to conduct semantic reasoning to enrich the existing data. Still using the Example-1 in clause 8.7.2.2 as an example, an oneM2M user (e.g., the owner of the Camera-11) may proactive trigger a semantic reasoning process over the semantic annotation of <Camera-11> (i.e., Fact-1 and Fact-2 as existing facts defined in 8.7.2) by using Feature-3 and RR-2. The semantic reasoning result (Inferred Fact-1, i.e., Camera-11 is-a ontologyB:VideoRecorder) is also a low-level semantic metadata about <Camera-11> and is a “long-term-effective” fact; therefore, such new/inferred fact can be further added/integrated into the semantic annotations of <Camera-11>. In other words, the existing facts now is “enriched or augmented” by the inferred fact. As a result, <Camera-11> can get more chance to be discovered by future semantic resource discovery operations. Another advantage from such enrichment is that future semantic resource discovery operations do not have to further trigger semantic reasoning in the background every time as supported by Feature-2, which helps reduce processing overhead and response delay. However, it is worth noting that it might not be applicable for integrating the inferred facts with existing facts in all the use cases. Taking the Example-2 in clause 8.7.2.2 as an example, the Inferred Fact-2 (i.e., “Camera-11 monitors-room-in MZ-1”) is relatively high-level knowledge, which may not be appropriate to be integrated with low-level semantic metadata (i.e., Fact-1 and Fact-2). In the meantime, since the hospital room allocation may get re-arranged from time to time, the Inferred Fact-2 may just be a “short-term-effective” fact. For instance, after a recent room re-allocation, Camera-11 does not monitor a room belonging to MZ-1 although Camera-11 is still located in Room-232 of Building-1 (i.e., Fact-1 and Fact-2 are still valid) but this room is now used for another purpose and then belongs to a different MZ (i.e., Inferred Fact-2 is no longer valid anymore and needs to be deleted). Therefore, it does not make sense to directly integrate such type of inferred fact or knowledge into the semantic annotations of massive cameras, otherwise it potentially leads to considerable annotation update overhead. Based on above discussion, tt can be seen that both Feature-2 and Feature-3 are the necessary features of SRF and each of them is to support different user cases respectively.

Overall, the general flow of Feature-3 is that oneM2M users (as originator) can send requests to certain receiver CSEs that has the reasoning capability. Accordingly, the receiver CSE will conduct a reasoning process by using the desired inputs (i.e., inputFS and RS) and produce the reasoning result and finally send the response back to the originator.

### -----------------------End of change 1-------------------------------------------