International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Y.4477

(11/2021)

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Internet of things and smart cities and communities – Frameworks, architectures and protocols

Framework for service interworking with device discovery and management in heterogeneous Internet of things environments

Recommendation ITU-T Y.4477

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Computing power networks	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	Y.3000–Y.3499
CLOUD COMPUTING	Y.3500–Y.3599
BIG DATA	Y.3600–Y.3799
QUANTUM KEY DISTRIBUTION NETWORKS	Y.3800–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	**Y.4400–Y.4549**
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T Y.4477

Framework for service interworking with device discovery and management in heterogeneous Internet of things environments

Summary

Recommendation ITU-T Y.4477 specifies a framework for service interworking with device discovery and management in heterogeneous Internet of things environments.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Y.4477	2021-11-29	20	11.1002/1000/14816

Keywords

Data management, device discovery, Internet of things (IoT), service interworking.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Page

Recommendation ITU-T Y.4477

Framework for service interworking with device discovery and management in heterogeneous Internet of things environments

1 Scope

This Recommendation specifies a framework for service interworking with device discovery and management in heterogeneous Internet of things (IoT) environments.

This Recommendation includes, for service interworking frameworks:

- overviews;
- functional requirements;
- functional architectures;
- procedures.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

None.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 Internet of things (IoT) [b-ITU-T Y.4000]: A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1 – Through the exploitation of identification, data capture, processing, and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2 – From a broader perspective, the IoT can be perceived as a vision with technological and societal implications.

3.1.2 service [b-ITU-T Y.101]: A structure set of capabilities intended to support applications.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 device discovery: The process of discovering devices to provide required services.

NOTE – Based on the description of "device and service discovery" in clause 8.3 of [b-ITU-T Y.4101].

3.2.2 service discovery: The process of discovering services to be provided by devices.

NOTE – Based on the description of "device and service discovery" in clause 8.3 of [b-ITU-T Y.4101].

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CM	Connection Manager
DB	Database
DR	Device Registry
ID	Identifier
IoT	Internet of Things
LD	Local Device
PM	Profile Manager
RIR	Registry Identifier Resolver
URL	Uniform Resource Locator

5 Conventions

None.

6 Overviews of service interworking

IoT service environments have been developed to support various IoT application domains (e.g., smart home, smart factory, smart building, smart farm and smart city) and many devices (e.g., smartphones, tablets, television sets, vehicles and home appliances). IoT services are also provided with thousands of IoT technologies; furthermore, the technologies can be heterogeneous. For this reason, interworking between IoT devices and services can give rise to difficulties despite both having the same objectives.

For instance, a smart home is a representative IoT service domain, many IoT services for the smart home have started in different ways. There are so many applications for smartphones for services in a smart home. All applications have similar service functions (e.g., remote switch on/off), but they cannot provide service functions for devices that do not have a technical or structural relationship with each other. In other words, a single application cannot provide all IoT services (e.g., controlling connected home appliances) for a smart home. This circumstance is caused by a fragmentation phenomenon of the IoT ecosystem. In addition, such fragmentation has produced heterogeneity of IoT devices, platforms, services and networks.

The heterogeneity in IoT ecosystems gives negative influences on interworking between two or more devices or services. If two home appliances that have the same IoT services (e.g., lighting) are produced by different vendors their services cannot interwork.

To provide interoperability between two heterogeneous IoT services from different providers, a process for transition between them is required by a device. This process of transition will be capable of discovering unknown devices, and managing information for interworking between them. The information will also include data models, protocols and interfaces. Furthermore, heterogeneous IoT services are caused by different IoT service platforms. To provide interoperability between the different service platforms, cross-platform functionality is also required.

This Recommendation specifies a framework of service interworking with device discovery and management in heterogeneous IoT environments with functional requirements, architectures and procedures.

7 Functional requirements of service framework for interworking

7.1 Device discovery for service interworking

Initiation of fundamental environments for device discovery is required in supporting service interworking of their frameworks. Therefore, the service interworking framework is required to:

– collect different device information to provide service interworking in heterogeneous IoT service environments;

– integrate device information in heterogeneous IoT service environments.

7.2 Support for cross-platform services

Device discovery among heterogeneous platforms is required to broadcast and offer device information. It also provides common functions for IoT service interworking. Therefore, the service interworking framework is required to:

– provide a device registration function for device discovery between different IoT service environments, as well as to register service-related information for optimizing service interworking without the involvement of a specific administrator.;

– coordinate different IoT device discovery mechanisms through cross-platform services;

– support an information (e.g., IoT device types) exchange function between different IoT services;

– collect service-related information from heterogeneous IoT service environments;

– support the advertisement of service-related information.

7.3 Data management for heterogeneous services

A service interworking framework requires capabilities for device information management, including mechanisms to collect the device profile, and to monitor device status due to the different properties of IoT services. Therefore, the service interworking framework is required to support:

– the collection of a device profile;

– the monitoring of the device status – by monitoring the status of devices, it can identify considerations for surveillance.

7.4 Identification for entities and device profiles

A service interworking framework consists of five different entities (described in clause 9) for communications and device profiles including service functions. The service framework is required to support identification:

– for all entities so that they can communicate with each other;

– of composite device profiles for service interworking.

7.5 Security for service discovery

It is required to provide security functions for service discovery. The data related to service discovery interworking must be secured, trusted and protected, so that unauthorized access needs to be prohibited. Therefore, the service framework is required to:

– protect against unauthorized illegal access – a modification process without a permission request must be forbidden;

– secure information management of all devices in local networks and entities in the cloud network.

8 Functional architectures for a service interworking framework

Figure 1 depicts functional architectures for service interworking frameworks in heterogeneous IoT environments.

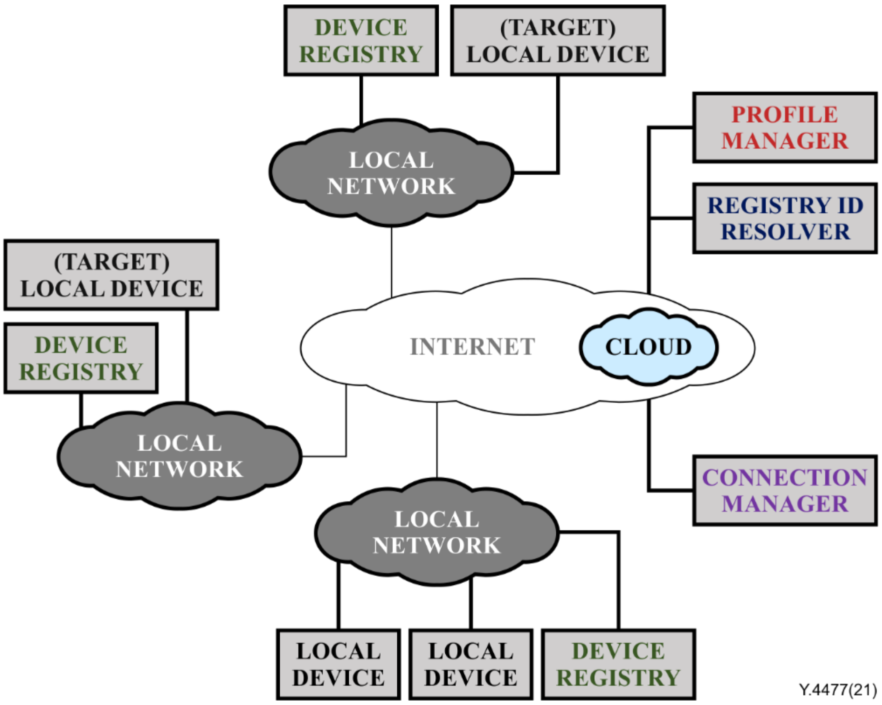

Figure 1 – Functional architectures of a service interworking framework

A service interworking framework consists of three types of centralized entity: the profile manager (PM); registry identifier (ID) resolver (RIR); and connection manager (CM) in the cloud network and two types of distributed entity, such as the device registry (DR) and local device (LD), in local networks. The Internet interconnects all entities in:

- local networks: DR, LD;
- a cloud network: PM, RIR and CM.

The entities in the cloud network are operated and managed by the same service provider. Moreover, the entities in local networks can be operated by different service providers.

The entities in local networks manage the service profiles of an LD for composite integrated services. The entities in cloud network resolution for IDs of device profiles manage information about device profiles. Clause 9 gives detailed information about the functionality of each entity.

9 Functionalities of entities for service interworking framework

9.1 Entities in cloud network

9.1.1 Profile manager

A PM has an objective to support the functionality of intelligent discovery for harmonized device profiles through a context-aware mechanism for composite integrated services. When the LDs appear in local networks, PM registers their information about device profiles in the database (DB) through registration procedures (see clause 10 for details). Furthermore, when an LD demands target device profiles for interworking through discovery procedures, a PM provides the stored information of device profiles to the LD. To do this, PM has two DBs (i.e., *DB_DEVICE_PROFILE* and

DB_SERVICE_PROFILE) exploited for matching service and device profiles. The following scheme shows data structures of the two DBs:

```
DATABASE: DEVICE_PROFILE

        • (TYPE_INT)        PROFILE_ID          # ID of Device Profile
        • (TYPE_STRING)     FUNCTION_NAME       # Name of Function
```

The *DB_DEVICE_PROFILE* is composed of two attributes, the ID of device profiles (data type: unsigned numeric) and names of functions (data type: string).

```
DATABASE: SERVICE_PROFILE
        • (TYPE_STRING)     SERVICE_NAME        # Name of Service
        • (TYPE_STRING)     FUNCTION_NAME       # Name of Function
```

The *DB_SERVICE_PROFILE* is composed of two attributes, the names of device profiles (data type: string) and names of functions (data type: string). It is assumed that records of the *DB_SERVICE_PROFILE* are fully registered without another registration procedure given in this Recommendation.

9.1.2 Registry identifier resolver

An RIR has an objective to provide connection information (e.g., a uniform resource locator (URL)) of registered DRs related to target LDs by resolving IDs of device profiles. RIR has a DB (i.e., *DB_REGISTRY*), and the following shows the data structures:

```
DATABASE: REGISTRY
        • (TYPE_INT)        REGISTRY_ID         # ID of Registry
        • (TYPE_STRING)     REGISTRY_URL        # URL of Registry
        • (TYPE_INT)        PROFILE_ID          # ID of Device Profile
```

The *DB_REGISTRY* is composed of three attributes, the ID of the registry (data type: unsigned numeric), the URL of the registry (data type: string), and the ID of device profiles (data type: unsigned numeric).

9.1.3 Connection manager

A CM has two objectives for monitoring connections that are available and controlling connection congestion. After discovery procedures, LDs start a process of connection and the updated information of connections available are registered to *DB_AVAILABLE_CONNECTION* of CM. CM has a DB (i.e., *DB_AVAILABLE_CONNECTION*), and the following scheme shows the data structures:

```
DATABASE: AVAILABLE_CONNECTION
        • (TYPE_INT)        PROFILE_ID          # ID of Device Profile
        • (TYPE_STRING)     FUNCTION_NAME       # Name of Function
        • (TYPE_BOOL)       AVAILABLE           # Availability for Next Connection
```

The *DB_AVAILABLE_CONNECTION* is composed of three attributes, the ID of device profiles (data type: unsigned numeric), name of function (data type: string), and availability of the next connection (data type: bool (TRUE/FALSE)).

9.2 Entities in local networks

9.2.1 Device registry

A DR has an objective to manage information about device profiles of LDs in a local network and to support the registration of the information to the entities in a cloud network. During discovery procedures, DRs provide information for connections to LDs in the same local network. Furthermore, DRs utilize a timer for sustaining up-to-date information. After a predefined time has passed, DRs check the status of LD whether they are active or not. DRs have a DB (i.e., *DB_DEVICE*), and the following shows the data structures:

```
DATABASE: DEVICE
        • (TYPE_INT)        DEVICE_ID       # ID of Local Device
        • (TYPE_STRING)     DEVICE_URL      # URL of Local Device
        • (TYPE_INT)        PROFILE_ID      # ID of Device Profile
        • (TYPE_INT)        TIMER           # Timer
```

The *DB_DEVICE* is composed of four attributes, ID of LD (data type: unsigned numeric), URL of LD (data type: string), the ID of device profiles (data type: unsigned numeric), and timer (data type: unsigned numeric).

9.2.2 Local device

LDs are described as devices that have device profiles for composite services with interworking in a local network. During registration procedures, information about LDs is registered in the DR in the same local network. Moreover, they also provide responses to connection requests from another LD after discovery procedures. LDs have a DB (i.e., *DB_DEVICE_PROFILE*), and the following scheme shows the data structures:

```
DATABASE: DEVICE PROFILE
        • (TYPE_STRING)     FUNC_NAME           # Name of Function
        • (TYPE_FUNC)       FUNC_TYPE           # Type of Function
        • (TYPE_INT)        REMAINING_RES       # Remaining Connection Resources
```

The *DB_DEVICE_PROFILE* is composed of three attributes, the name of functions (data type: string), type of functions (data type: *TYPE_FUNC*), and the remaining connection resources (data type: unsigned numeric). The data type, *TYPE_FUNC* is described in the following scheme:

```
DATA TYPE: TYPE_FUNC
        • TYPE_SENSOR           # Type of Sensors
        • TYPE_ACTUATOR         # Type of Actuators
        • TYPE_PROCESSOR        # Type of Processor
```

NOTE – Attributes with the same name in different DBs are of the same data type, but the contents can differ in each DB according to which entity is corresponding.

10 Procedures for service interworking framework

10.1 Registration procedures for device profiles

10.1.1 De/registration between local devices and a device registry in local networks

In the first stage, Figure 2 depicts registration procedures between LDs and DR in a local network. A DR sends a broadcast message, ***DISCOVERY_REQ(NONE)*** to discover unknown LDs within the same subnet. All LDs that receive the broadcast message reply with unicast messages, ***DISCOVERY_REP(DEVICE_ID)***, to the DR. The replies include IDs of LDs. The DR registers new records including the IDs and URLs of LDs in *DB_DEVICE* and sends a unicast message, ***PROFILE_REQ(DEVICE_ID)***, to each LD to request its device profile. Each LD replies with a unicast message, ***PROFILE_REP(DEVICE_PROFILE)***, to the DR. The message includes all device profiles of the LD. The device profile includes a tentative ID of the device profiles and a name of the function. LDs can have one ID of device profiles, and the ID can correspond to one or more functions. That is, LDs can have one or more functions. The DR receives all messages from LDs and updates the tentative ID and the corresponding functions in the existing records in the *DB_DEVICE*. Furthermore, a timer with a predefined time is set and included in each record.

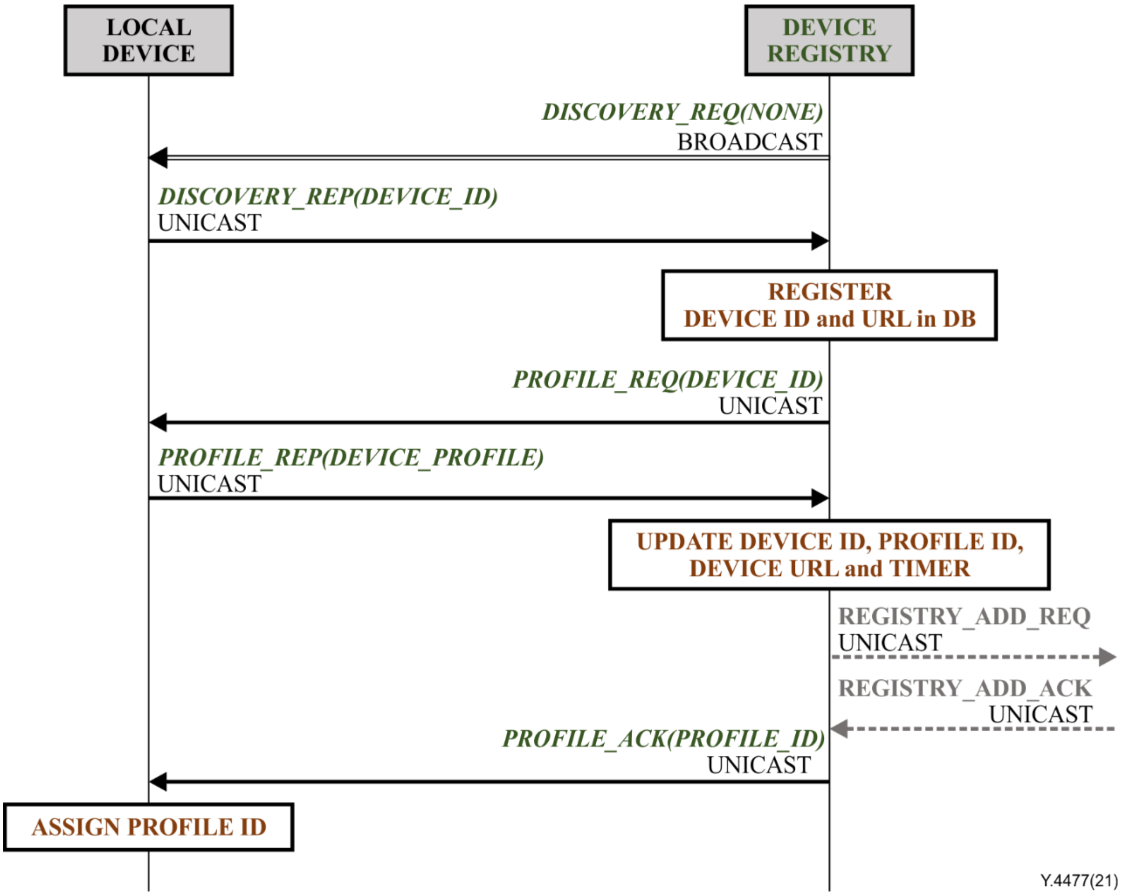

Figure 2 – Registration procedures between local devices and a device registry

The DR then starts the registration procedures of the second stage on the cloud network. On completion, a regular ID of the device profile is established. The DR finally updates the regular ID of the device profile in the *DB_DEVICE* and sends a confirmation message, ***PROFILE_ACK(PROFILE_ID)***, to each LD. The LDs finally receive the assigned regular ID of the device profile.

Figure 3 depicts deregistration procedures between LDs and a DR when the LDs disappear or cannot interwork. The DR regularly sends a unicast message, ***DISCOVERY_REQ(NONE)***, to the corresponding LD whenever a timer is out. When the LD receives a unicast message, the LD decides whether to finish the interworking. If finished, the LD sends a unicast message, ***PROFILE_RST(PROFILE_ID)***, to the DR for deregistration. If unfinished, the LD replies with a confirmation message, ***DISCOVERY_ACK(NONE)***, to the DR.

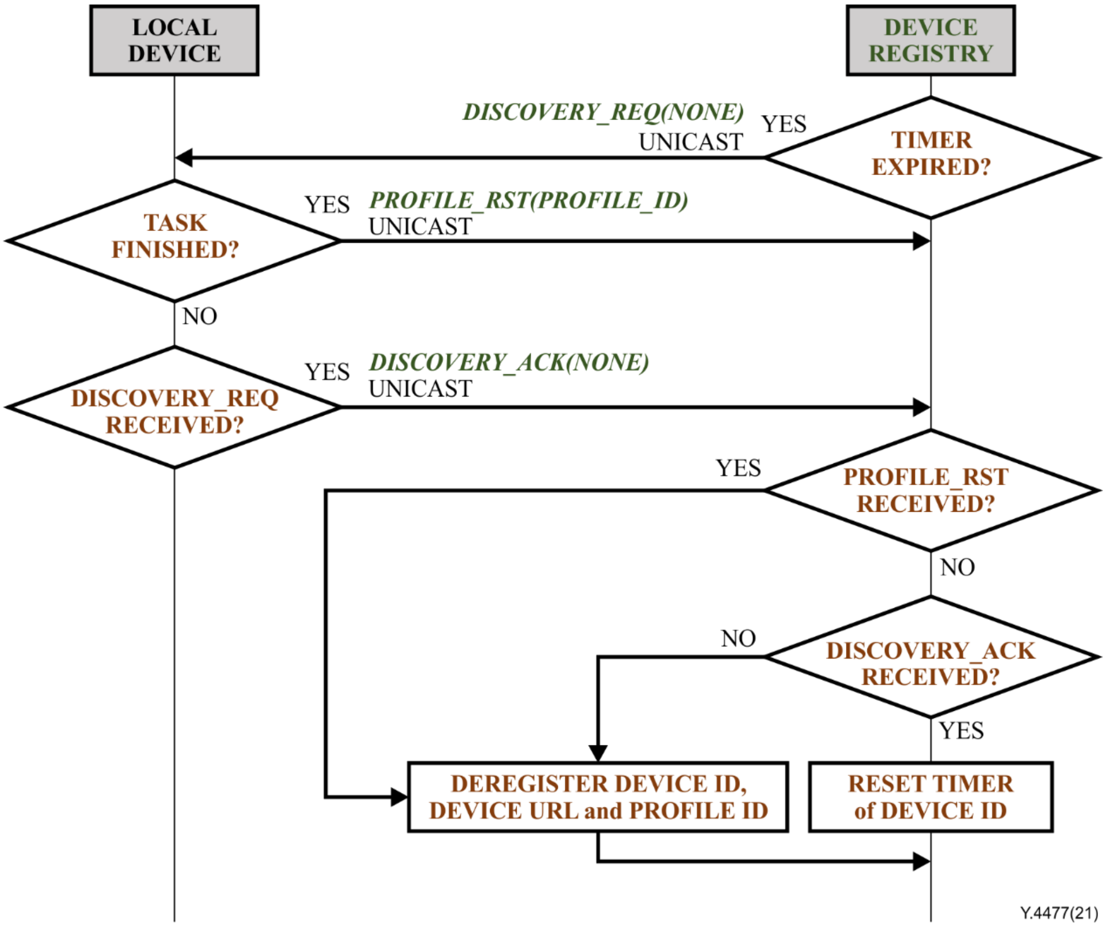

Figure 3 – Deregistration procedures between local devices and a device registry

When the DR receives the message ***PROFILE_RST(PROFILE_ID)*** or does not receive the reply ***DISCOVERY_ACK(NONE)***, the corresponding record in the ID of the device profile is finally deregistered (removed) in the *DB_DEVICE*. Otherwise, the DR resets the timer with the predefined time of the corresponding record. After that, the DR conducts deregistration procedures with entities in the cloud network.

10.1.2 De/registration between device registries and entities in cloud network

Figure 4 depicts registration and deregistration procedures with the entities (i.e., DRs, RIR and PM) in a cloud network. As described in clause 10.1.1, when a new device joins interworking, it gets registered in the *DB_DEVICE* of a DR, which then sends a message, ***REGISTRY_ADD_REQ(DEVICE_PROFILE)***, with the new device profile to an RIR. The RIR creates and assigns a regular ID of the received device profile, and the RIR registers a new record with an ID and a URL of the DR and the regular ID of the device profile in *DB_REGISTRY*. After that, the RIR sends a message, ***PROFILE_ADD_REQ(PROFILE_ID, DEVICE_PROFILE)***, to a PM. This message includes the device profile and its ID. When the PM receives the message, the PM registers a new record including the device profile and its ID in *DB_DEVICE_PROFILE*. The PM replies with a confirmation message, ***PROFILE_ADD_ACK(PROFILE_ID)***, to the RIR to notify completion of registration. After receiving the message from the PM, the RIR sends a message, ***REGISTRY_ADD_ACK(PROFILE_ID)***, to the DR to notify completion of the registration.

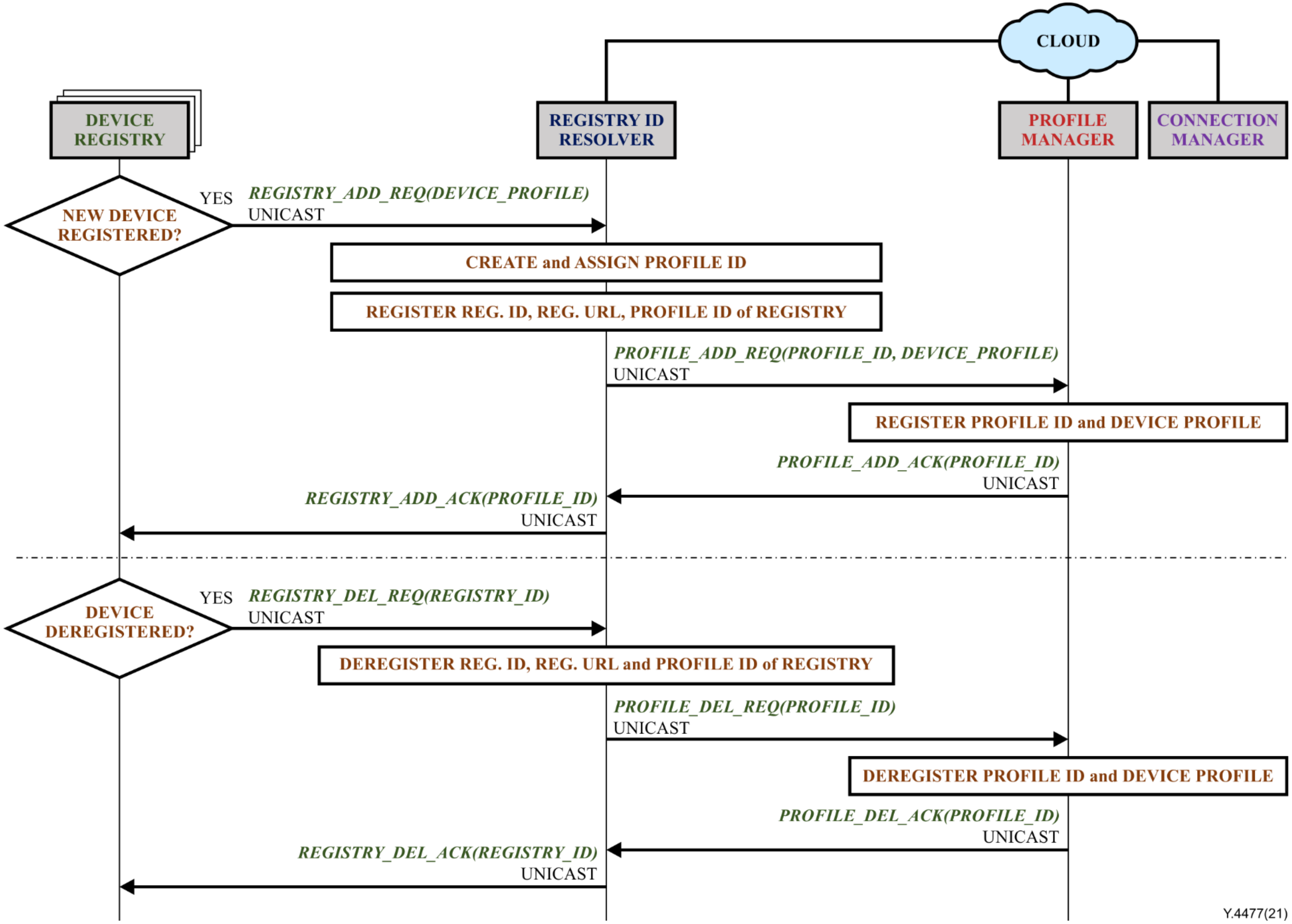
CLOUD
DEVICE REGISTRY
REGISTRY ID RESOLVER
PROFILE MANAGER
CONNECTION MANAGER
NEW DEVICE REGISTERED?
YES
REGISTRY_ADD_REQ(DEVICE_PROFILE)
UNICAST
CREATE and ASSIGN PROFILE ID
REGISTER REG. ID, REG. URL, PROFILE ID of REGISTRY
PROFILE_ADD_REQ(PROFILE_ID, DEVICE_PROFILE)
UNICAST
REGISTER PROFILE ID and DEVICE PROFILE
PROFILE_ADD_ACK(PROFILE_ID)
UNICAST
REGISTRY_ADD_ACK(PROFILE_ID)
UNICAST
DEVICE DEREGISTERED?
YES
REGISTRY_DEL_REQ(REGISTRY_ID)
UNICAST
DEREGISTER REG. ID, REG. URL and PROFILE ID of REGISTRY
PROFILE_DEL_REQ(PROFILE_ID)
UNICAST
DEREGISTER PROFILE ID and DEVICE PROFILE
PROFILE_DEL_ACK(PROFILE_ID)
UNICAST
REGISTRY_DEL_ACK(REGISTRY_ID)
UNICAST
Y.4477(21)

Figure 4 – De/registration procedures with device registries, registry identifier resolver and profile manager

When a device profile is deregistered in *DB_DEVICE* of a DR, the DR sends a message, ***REGISTRY_DEL_REQ(REGISTRY_ID)***, with an ID of the DR to an RIR. The RIR then deregisters (removes) the corresponding record of the DR in *DB_REGISTRY*. After that, the RIR sends a message, ***PROFILE_DEL_REQ(PROFILE_ID)***, to a PM. This message includes the ID of the device profile. When the PM receives the message, the PM deregisters the corresponding record of the device profile in *DB_DEVICE_PROFILE*. The PM replies with a message, ***PROFILE_DEL_ACK(PROFILE_ID)***, to the RIR to notify completion of deregistration. After receiving the message from the PM, the RIR sends a confirmation message, ***REGISTRY_DEL_ACK(REGISTRY_ID)***, to the DR to notify completion of deregistration.

10.2 Discovery procedures

Figure 5 depicts the discovery procedure by which an LD finds out about device profiles of other (target) LDs for service interworking. To find a required device profile, an LD sends a message, ***DISCOVERY_PROFILE_REQ(DEVICE_PROFILE)***, including its own device profile to a DR. The DR adds information (the ID of the device profile) to the message, ***DISCOVERY_PROFILE_REQ(PROFILE_ID, DEVICE_PROFILE)***, and sends it to an RIR. The RIR forwards the message to a PM. The PM executes a process to search for related device profiles, used for service interworking. The process is based on detailed procedures for "***SEARCH MATCHED SERVICES AND PROFILE***" with a device profile as an input value (see Annex A). The process is repeated until all related device profiles are found in the *DB_DEVICE_PROFILE*, matched with *DB_SERVICE_PROFILE*.

Whenever a related device profile is found from the *DB_DEVICE_PROFILE*, the PM requests verification for connection available to a CM with a message ***VERIFY_CONNECT_REQ((TARGET DEVICE) PROFILE_ID)***. The CM verifies a result (output: TRUE/FALSE) from the *DB_AVAILABLE_CONNECTION* with the received ID of the device profile by using detailed procedures for "***CHECK AVAILABLE CONNECTION***" (see Annex B). After this process, the CM replies with the result to the PM with a message, ***VERIFY_CONNECT_REP((TARGET DEVICE) PROFILE_ID, #AVAILABLE)***. If it is 'TRUE', the PM replies with the information (name of service, name of function, the ID of the device profile) of a target LD to the RIR with a message, ***DISCOVERY_PROFILE_REP(#SERVICE, #FUNC, DEV. PROFILE_ID)***. The RIR searches a URL of a DR in the target side. The information (a URL of a DR, name of service, name of the function, ID of device profile) related to the target LD is forwarded to the LD through the DR with a reply message, ***DISCOVERY_PROFILE_REP (REGISTRY_URL, #SERVICE, #FUNC, PROFILE_ID)***.

After that, the LD requests information about a target LD from the DR on the target side with the received URL with a message, ***DISCOVERY_DEVICE_REQ(PROFILE_ID)***. The DR on the target side replies with information about the target LD with a message, ***DISCOVERY_DEVICE_REP(DEVICE_URL)***. The LD finally obtains all information to connect and interwork with the target LD.

The last stage then initiates connection status registration procedures by device connection (see clause 10.3).

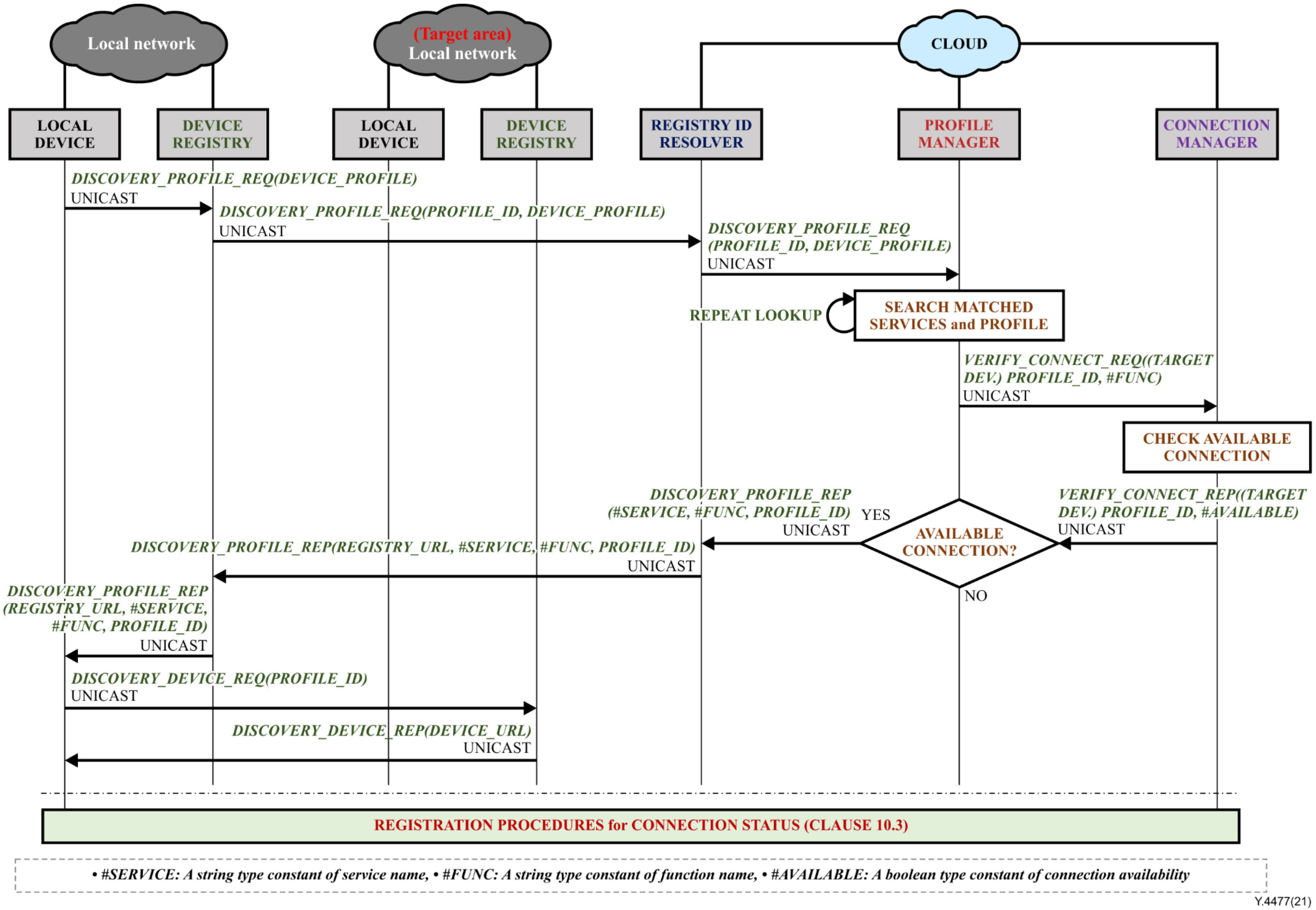
Local network
(Target area)
Local network
CLOUD
LOCAL DEVICE
DEVICE REGISTRY
LOCAL DEVICE
DEVICE REGISTRY
REGISTRY ID RESOLVER
PROFILE MANAGER
CONNECTION MANAGER
DISCOVERY_PROFILE_REQ(DEVICE_PROFILE)
UNICAST
DISCOVERY_PROFILE_REQ(PROFILE_ID, DEVICE_PROFILE)
UNICAST
DISCOVERY_PROFILE_REQ (PROFILE_ID, DEVICE_PROFILE)
UNICAST
REPEAT LOOKUP
SEARCH MATCHED SERVICES and PROFILE
VERIFY_CONNECT_REQ((TARGET DEV.) PROFILE_ID, #FUNC)
UNICAST
CHECK AVAILABLE CONNECTION
DISCOVERY_PROFILE_REP (#SERVICE, #FUNC, PROFILE_ID)
UNICAST
YES
AVAILABLE CONNECTION?
NO
VERIFY_CONNECT_REP((TARGET DEV.) PROFILE_ID, #AVAILABLE)
UNICAST
DISCOVERY_PROFILE_REP(REGISTRY_URL, #SERVICE, #FUNC, PROFILE_ID)
UNICAST
DISCOVERY_PROFILE_REP (REGISTRY_URL, #SERVICE, #FUNC, PROFILE_ID)
UNICAST
DISCOVERY_DEVICE_REQ(PROFILE_ID)
UNICAST
DISCOVERY_DEVICE_REP(DEVICE_URL)
UNICAST
REGISTRATION PROCEDURES for CONNECTION STATUS (CLAUSE 10.3)
• #SERVICE: A string type constant of service name, • #FUNC: A string type constant of function name, • #AVAILABLE: A boolean type constant of connection availability
Y.4477(21)

Figure 5 – Discovery procedures

10.3 Registration procedures for connection status

Figure 6 depicts registration procedures for connection status when two LDs have a new connection for service interworking. An LD requests a function for service interworking to the LD in a target side with a message, ***DEVICE_FUNC_REQ(#FUNC)***. When the LD receives a message, ***DEVICE_FUNC_ACK(#FUNC)***, from the LD in the target side, the two LDs conduct service interworking. After being connected, each LD decides to accept more connections available from the remaining functional resources. Each of them sends the result (output: TRUE/FALSE) to the CM with a message, ***PROFILE_AVAILABLE_REG_REQ(PROFILE_ID, #FUNC, #AVAILABLE)***. The CM registers a new record (or updates the existing record) about the result in *DB_AVAILABLE_CONNECTION*. After that, the CM replies with a confirmation message, ***PROFILE_AVAILABLE_REG_ACK (PROFILE_ID)***, to each LD.

11 Security considerations

Security considerations for service discovery are critical. The security considerations between service discovery procedures should be established for interworking.

The items in the following list are considerations for service security.

- Preservation of service discovery contents. The access management of service contents uses an authentication process (e.g., encryption method or specific setting). Service security is necessary to certify service interworking.
- Frequent reporting of security consequences. Service security should be considered for automatic service discovery.
- Inspection of surveillance functions (e.g., range of security procedures, and monitoring methods).

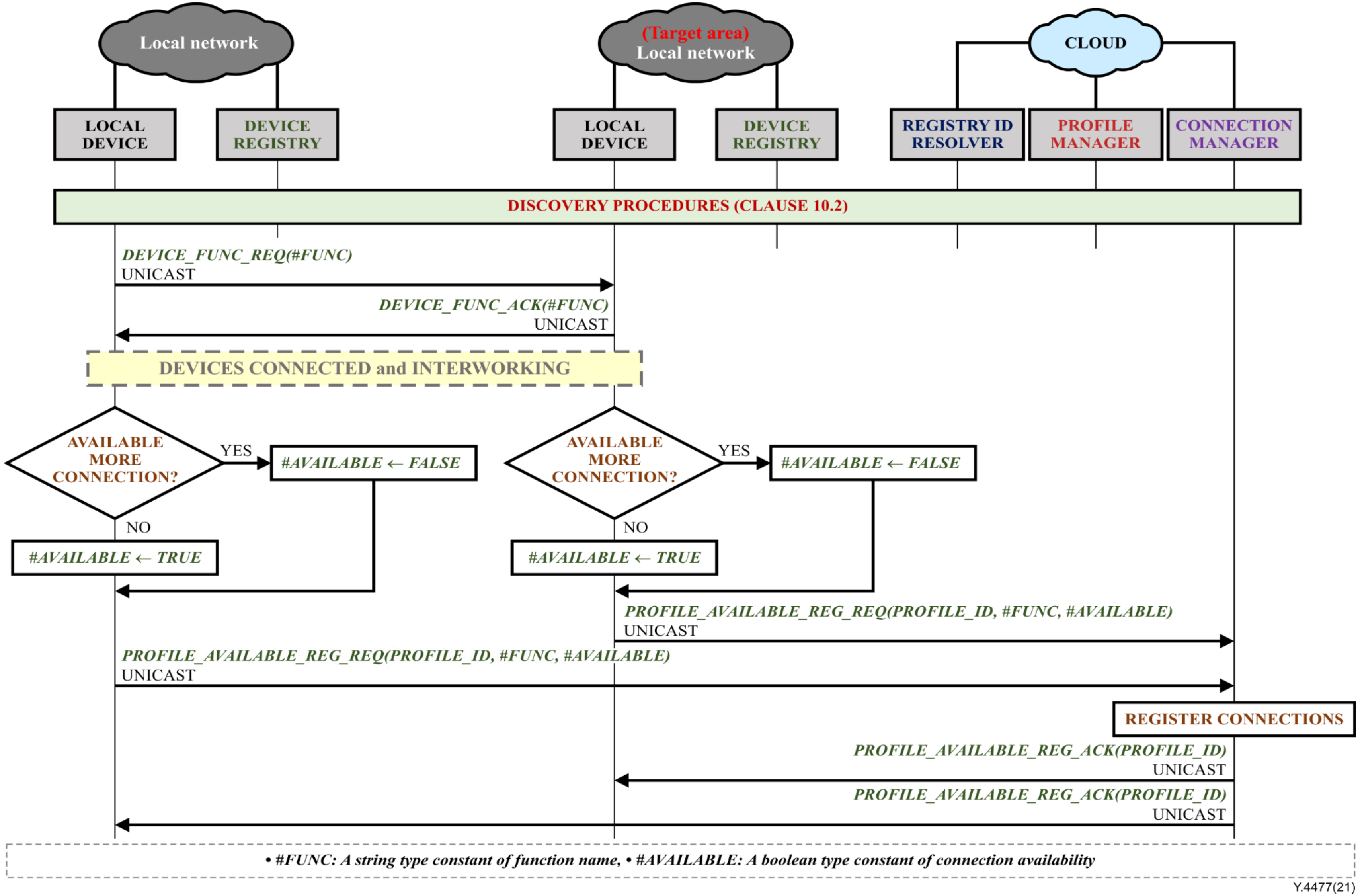
Local network
(Target area)
Local network
CLOUD
LOCAL DEVICE
DEVICE REGISTRY
LOCAL DEVICE
DEVICE REGISTRY
REGISTRY ID RESOLVER
PROFILE MANAGER
CONNECTION MANAGER
DISCOVERY PROCEDURES (CLAUSE 10.2)
DEVICE_FUNC_REQ(#FUNC)
UNICAST
DEVICE_FUNC_ACK(#FUNC)
UNICAST
DEVICES CONNECTED and INTERWORKING
AVAILABLE MORE CONNECTION?
YES
#AVAILABLE ← FALSE
NO
#AVAILABLE ← TRUE
AVAILABLE MORE CONNECTION?
YES
#AVAILABLE ← FALSE
NO
#AVAILABLE ← TRUE
PROFILE_AVAILABLE_REG_REQ(PROFILE_ID, #FUNC, #AVAILABLE)
UNICAST
PROFILE_AVAILABLE_REG_REQ(PROFILE_ID, #FUNC, #AVAILABLE)
UNICAST
REGISTER CONNECTIONS
PROFILE_AVAILABLE_REG_ACK(PROFILE_ID)
UNICAST
PROFILE_AVAILABLE_REG_ACK(PROFILE_ID)
UNICAST
• #FUNC: A string type constant of function name, • #AVAILABLE: A boolean type constant of connection availability
Y.4477(21)

Figure 6 – Registration procedures for connection status

Annex A

Detailed procedures for "SEARCH MATCHED SERVICES AND PROFILE"

(This annex forms an integral part of this Recommendation.)

This procedure for SEARCH MATCHED SERVICES AND PROFILE is conducted when PM receives the message, ***DISCOVERY_PROFILE_REQ(PROFILE_ID, DEVICE_PROFILE)***, from RIR as discussed in clause 10.2.

In the beginning stage, three temporary variables are required. A value, *DEVICE_PROFILE*, in the message is stored in the first temporary variable, TMP_PF, and the functions in the *DEVICE_PROFILE* are stored in the second temporary variable, TMP_FUNC. In addition, functions of the DB, ***DB_SERVICE_PROFILE*** are stored in the third temporary variable, TMP_SER.

In the next stage, to collect possible matched predefined service profiles with device profiles of the LD, whenever there is a number, the *#FUNC* of the variable, the TMP_SER, which is equivariant to the variable, and the TMP_FUNC, a tuple value (TMP_FUNC and TMP_SER) is stored in the form of a variable in a type of list, LR_SER. However, the variable LR_SER does not store the redundant values.

In the last stage, to discover device profiles of target LDs, every value, *#FUNC* of variable, TMP_PF is matched to the values, *#FUNC* of the variable, LR_SER. Whenever the two values, *#FUNC* of the different variables are equivariant, the message, ***VERIFY_CONNECT_REQ(PROFILE_ID, #FUNC)***, is sent to the CM. The CM verifies whether the LD is acceptable for another connection by using detailed procedures for CHECK AVAILABLE CONNECTION given in Annex B. If another connection is acceptable, a message, ***DISCOVERY_PROFILE_REP(#SERVICE, #FUNC, PROFILE_ID)***, is sent back to the PM.

See Figure A.1.

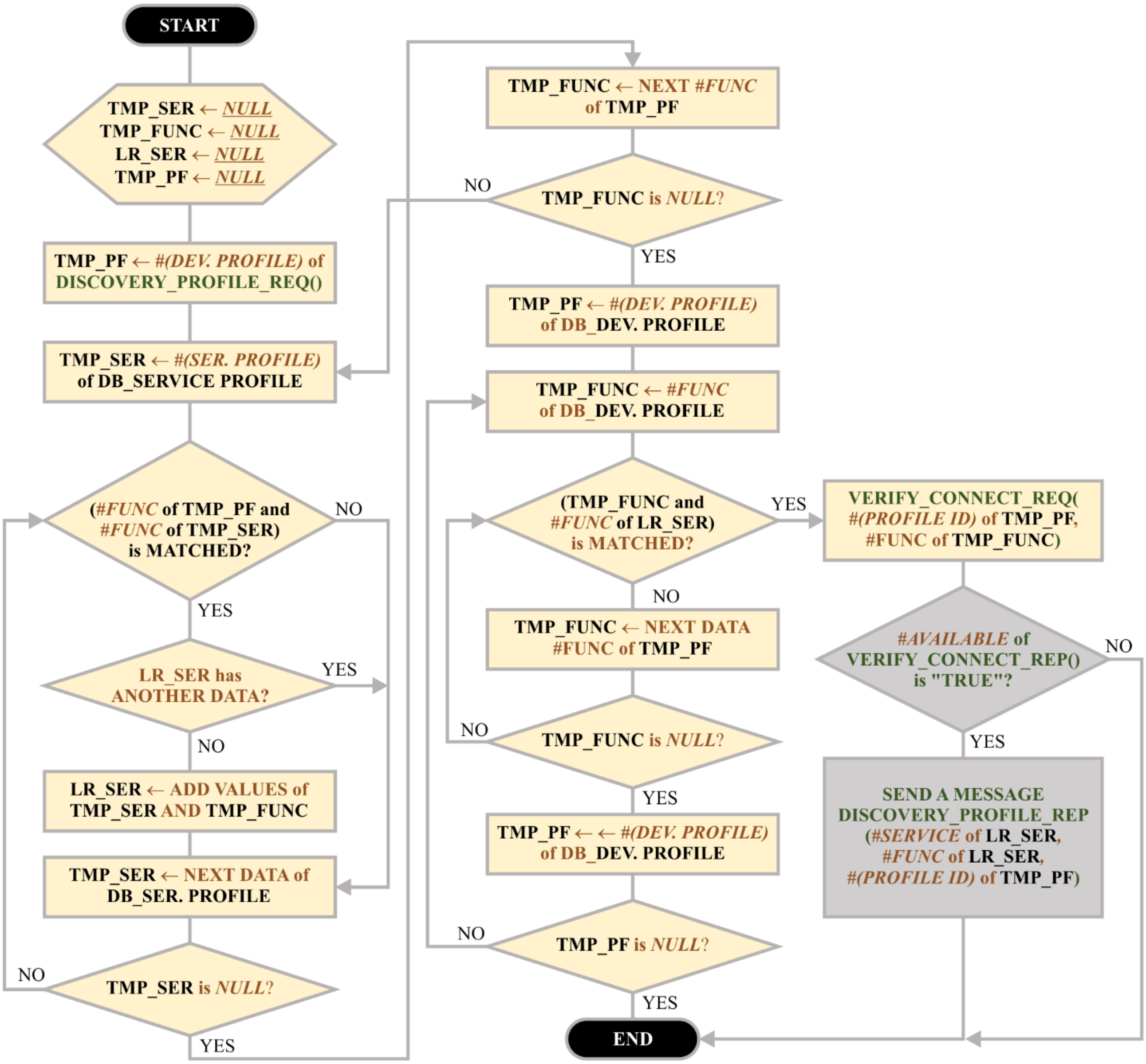
START
TMP_SER ← NULL
TMP_FUNC ← NULL
LR_SER ← NULL
TMP_PF ← NULL
TMP_PF ← #(DEV. PROFILE) of DISCOVERY_PROFILE_REQ()
TMP_SER ← #(SER. PROFILE) of DB_SERVICE PROFILE
(#FUNC of TMP_PF and #FUNC of TMP_SER) is MATCHED?
NO
YES
LR_SER has ANOTHER DATA?
YES
NO
LR_SER ← ADD VALUES of TMP_SER AND TMP_FUNC
TMP_SER ← NEXT DATA of DB_SER. PROFILE
NO
TMP_SER is NULL?
YES
TMP_FUNC ← NEXT #FUNC of TMP_PF
NO
TMP_FUNC is NULL?
YES
TMP_PF ← #(DEV. PROFILE) of DB_DEV. PROFILE
TMP_FUNC ← #FUNC of DB_DEV. PROFILE
(TMP_FUNC and #FUNC of LR_SER) is MATCHED?
YES
NO
TMP_FUNC ← NEXT DATA #FUNC of TMP_PF
NO
TMP_FUNC is NULL?
YES
TMP_PF ← ← #(DEV. PROFILE) of DB_DEV. PROFILE
NO
TMP_PF is NULL?
YES
END
VERIFY_CONNECT_REQ(#(PROFILE ID) of TMP_PF, #FUNC of TMP_FUNC)
#AVAILABLE of VERIFY_CONNECT_REP() is "TRUE"?
NO
YES
SEND A MESSAGE DISCOVERY_PROFILE_REP (#SERVICE of LR_SER, #FUNC of LR_SER, #(PROFILE ID) of TMP_PF)

Figure A.1 – Detailed procedures for SEARCH MATCHED SERVICES AND PROFILE

Annex B

Detailed procedures for "CHECK AVAILABLE CONNECTION"

(This annex forms an integral part of this Recommendation.)

To verify whether the target LD is acceptable for another connection, detailed procedures for CHECK AVAILABLE CONNECTION are conducted when the CM receives the message, ***VERIFY_CONNECT_REQ(PROFILE_ID, #FUNC)***, from the PM as discussed in clause 10.2.

The ***DB_AVAILABLE_CONNECTION***, which is handled by the CM, includes two types of information, (*PROFILE_ID*, *#FUNC*). If there is no information registered in the ***DB_AVAILABLE_CONNECTION***, the LDs are neither discovered nor registered. Secondly, the CM also checks whether any of the LDs are discovered. If discovered, a value, *#AVAILABLE*, of the matched LD in ***DB_AVAILABLE_CONNECTION*** is returned to the PM with a message, ***VERIFY_CONNECT_REP(PROFILE_ID, #AVAILABLE)***.

See Figure B.1.

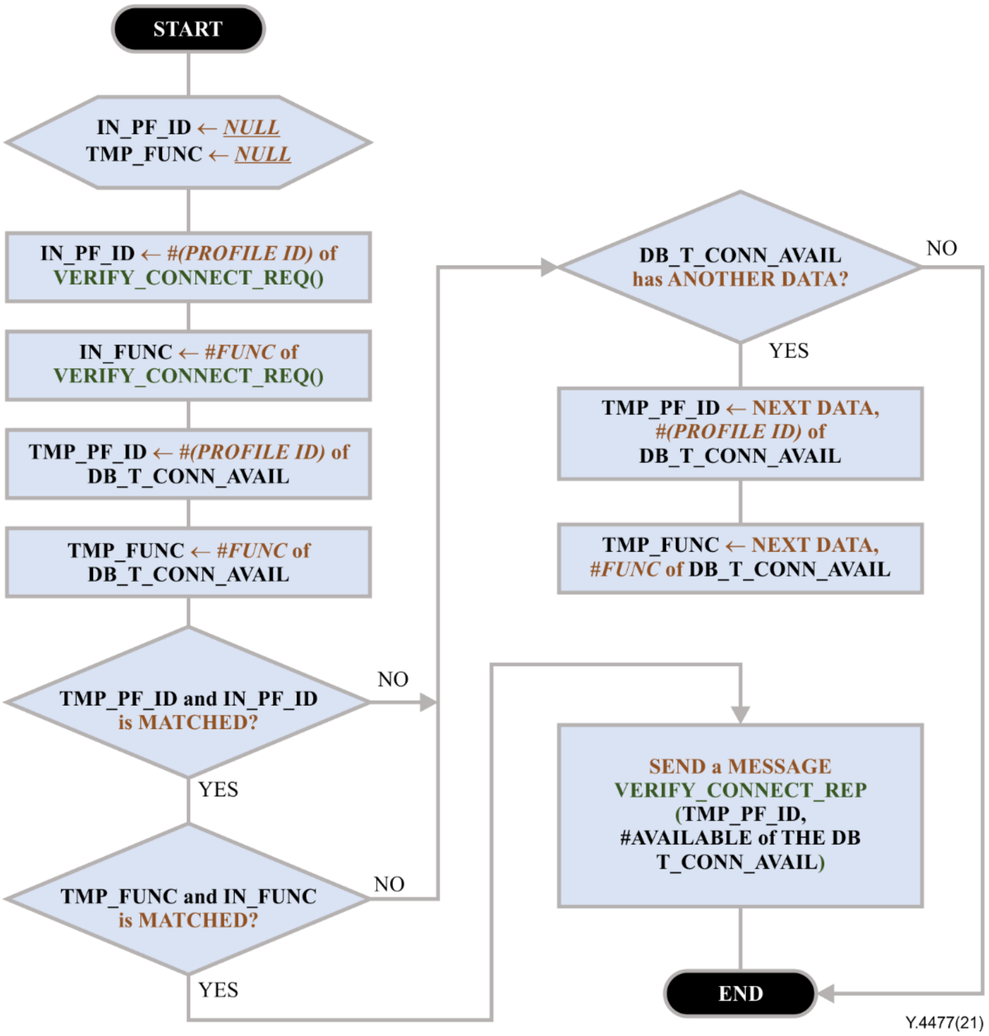
START
IN_PF_ID ← NULL
TMP_FUNC ← NULL
IN_PF_ID ← #(PROFILE ID) of VERIFY_CONNECT_REQ()
IN_FUNC ← #FUNC of VERIFY_CONNECT_REQ()
TMP_PF_ID ← #(PROFILE ID) of DB_T_CONN_AVAIL
TMP_FUNC ← #FUNC of DB_T_CONN_AVAIL
TMP_PF_ID and IN_PF_ID is MATCHED?
NO
YES
TMP_FUNC and IN_FUNC is MATCHED?
NO
YES
DB_T_CONN_AVAIL has ANOTHER DATA?
NO
YES
TMP_PF_ID ← NEXT DATA, #(PROFILE ID) of DB_T_CONN_AVAIL
TMP_FUNC ← NEXT DATA, #FUNC of DB_T_CONN_AVAIL
SEND a MESSAGE VERIFY_CONNECT_REP (TMP_PF_ID, #AVAILABLE of THE DB T_CONN_AVAIL)
END
Y.4477(21)

Figure B.1 – Detailed procedures for CHECK AVAILABLE CONNECTION

Bibliography

[b-ITU-T Y.101] Recommendation ITU-T Y.101 (2000), *Global Information Infrastructure terminology: Terms and definitions*.

[b-ITU-T Y.4000] Recommendation ITU-T Y.4000/Y.2060 (2012), *Overview of the Internet of things*.

[b-ITU-T Y.4101] Recommendation ITU-T Y.4101/Y.2067 (2017), *Common requirements and capabilities of a gateway for Internet of things applications.*

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2021