ecma INTERNATIONAL

Standard ECMA-412

3rd Edition / June 2019

Framework for distributed real-time access systems

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

Contents

Page

Introduction

Technology for real-time access control is widely used in many situations such as entrance gates of facilities and service access control systems. Membership and settlement services also benefit from real-time access control systems connected via networks and using database information.

Sophisticated cloud, virtualisation, database, networking technology and services and the evolution of authentication technology such as biometrics, NFC, QR codes used in distributed and modular access control systems enable previously underserved users and operators to innovate around new use cases.

Taking into account the many technologies, this Standard specifies the reference model and common control functions. It gives direction for ongoing innovation and development of technology and system integration of distributed real-time access control system.

The 2nd edition of the Standard introduced new functionalities on performance management mechanisms. Performance management mechanisms allow an access system to be evaluated for performance by using specific elements and metrics. This edition also provides a number of editorial improvements and clarifications to the text of the Standard.

NOTE In the 1st edition the title of the Standard was access systems.

This 3rd edition is fully aligned with ISO/IEC 20933:2019.

This Ecma Standard was developed by Technical Committee 51 and was adopted by the General Assembly of June 2019.

Framework for distributed real-time access systems

1 Scope

This Standard specifies a framework for a distributed real-time access system. It includes:

1) an ID triggered modular system architecture, the functions of the modules, the semantics of messages those modules exchange, and elements of messages;

2) the system behaviour from the time it receives an access request until the time it sends the result along with the sequence;

3) performance measurement mechanisms using a time stamping function that can be employed for the evaluation of the system.

2 Normative references

There are no normative references in this Standard.

3 Terms and definitions

For the purposes of this Standard, the following terms, definitions and acronyms apply.

3.1
Accessor
someone or something that interacts with the access system

3.2
access-ID
identifier in an Access request

3.3
access-ID-obtained-time
time when an Access-point module obtains an access-ID

3.4
access-point-ID
identifier of an Access-point module

3.5
Access-request
request trigger of processing for access system

3.6
Distributed real-time access system
data processing system distributed in the network which is activated by an access request and completed when the processing result accepts or denies that request within a reasonable period of time

3.7
Final-Result-Notification
notification of the final result of a transaction

3.8
function-ID
identifier of function

3.9
Policy-getter
message to request the Policy module to set the rules

3.10
Policy-setter
message to set the rules to the RED module

3.11
Processing-request
request to execute a function

3.12
Processing-response
response to a Processing-request

3.13
RED
Rule Evaluation and Dispatching

3.14
receivedTime
time when a module receives a request from another module

3.15
Retrieve-request
request to retrieve data from storage

3.16
Retrieve-response
response to a Retrieve-request

3.17
rule-ID
identifier of rules

3.18
sendingTime
time when a module sends a response or a Transaction-start-request to another module

3.19
Store-request
request to store data to storage

3.20
Store-response
response to a Store-request

3.21
Time-stamp-Notification
notification to provide time stamp information

3.22
transaction-ID
identifier of a transaction

3.23
Transaction-start-request
request to initiate a transaction

4 Conformance

Conformant access systems progress transactions by interpreting the applicable rules. Conformant modules implement the requests on their interfaces, the corresponding responses and time stamping as specified herein.

5 Overview

This clause is an overview of the system model and the functions of a distributed real-time access system.

The access system consists of 5 modules "Access-point, Policy, Processing, RED and Storage" and 4 interfaces "Access-interface, Policy-interface, Processing-interface and Storage-interface". There are also 2 external interfaces “In” and “Out”.

The access system model is shown in Figure 1.

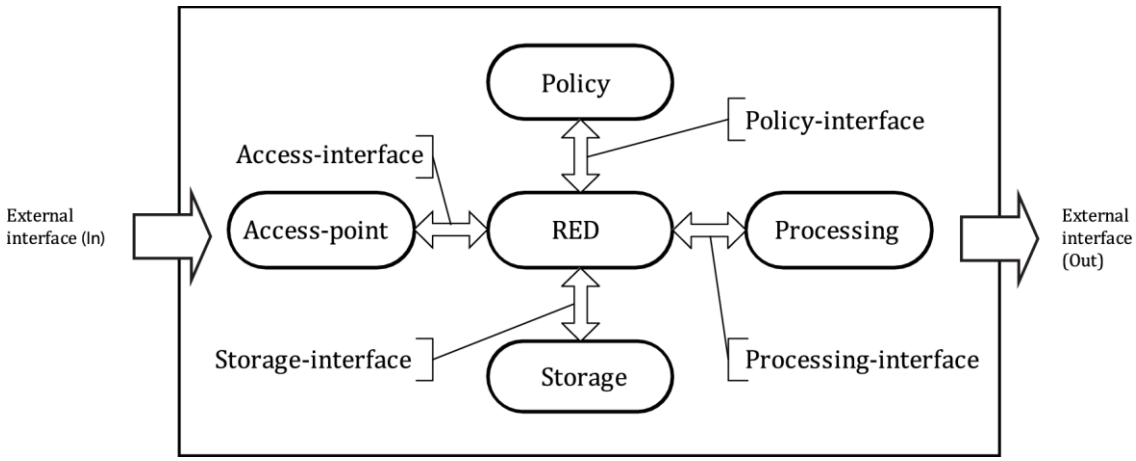

Figure 1 — Access system model

The access system starts a transaction triggered by an Access ID which is included in Access request from the Accessor through the external interface (In). After the necessary process, the access system completes the transaction by sending the final result to the receiver through the other external interface (Out).

The access system has a mechanism, the time stamp function, to measure processing time for the evaluation of the access system performance.

6 Transaction

A transaction is a suite of functions and message exchanges to generate a final result and send it to a receiver. A transaction starts from the time an access system receives an access request and completes after it sends the result.

When an Access-request is received by the Access-point module, a transaction proceeds to a generated state. In the generated state, the Access-point module generates a transaction-ID which identifies a transaction. The transaction_ID is created based on an activated access-ID. The Access-point module sends Transaction-start-request with the transaction-ID to the RED module.

After sending a Transaction-start-request, a transaction proceeds to an on-going state. At the on-going state, the RED module interprets the rules set by the Policy module. According to the result of the interpretation, the RED module sends request messages to the Processing or Storage module. Upon receiving a request message, the Processing module and the Storage module send response messages to the RED module. The RED module interprets the rules again. The RED module repeats the above procedure until the final result is decided based on rules and sends a final result (Final-Result-Notification) to the receiver through the external interface (Out).

After sending the final result, the transaction proceeds to a completed state. When a transaction is completed, the usage of the access-ID is also completed. An example of message sequence is shown in Annex A.

The state machine of a transaction is shown in Figure 2.

NOTE 1 access-ID is not defined in this Standard and is usually managed by a service provider. The life cycle and generation of an access-ID is not in the scope of this Standard.

NOTE 2 This behaviour of a transaction described above is for a transaction under stable condition when a response based on a request during a transaction is received within a reasonable period of time.

In the case of a system fault, such as power loss, network failure, or module malfunction when no response is received within a reasonable period of time, this Standard does not define any exceptional system management rules. However, the rules for providing such system failure, such as stopping a transaction, resetting the system, or making a re-access request to the Accessor, should be provided in the actual system.

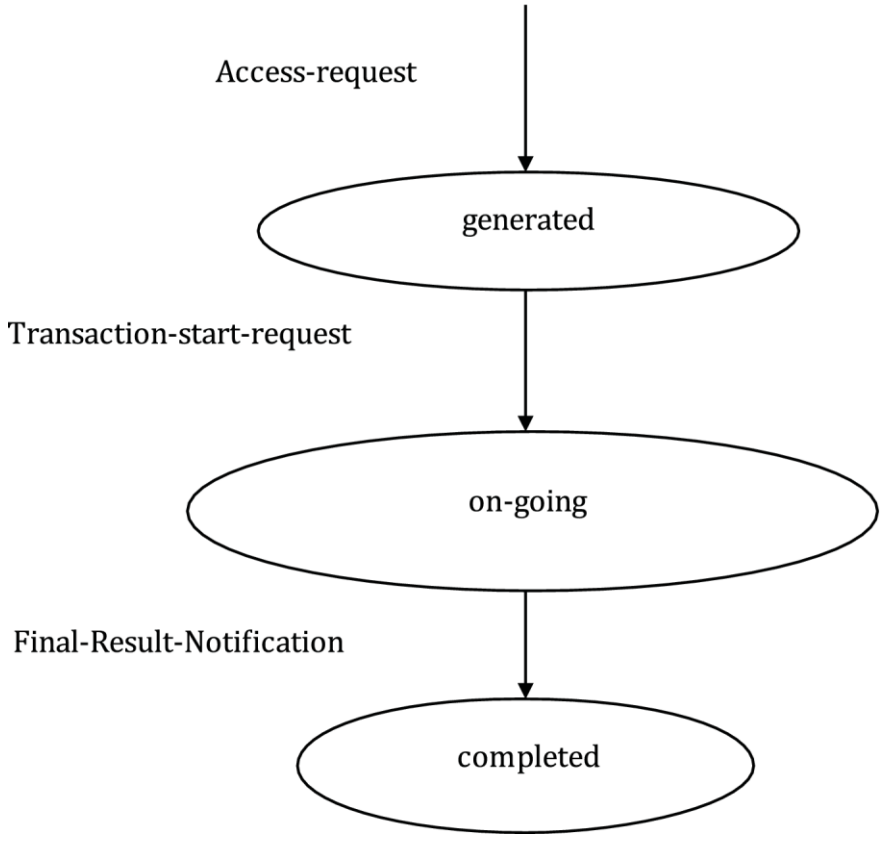

Figure 2 — Transaction state machine

The rules are composed of procedural steps and branch steps to determine exchanges of messages. Figure 3 illustrates a procedural step and Figure 4 illustrates a branch step. A procedural step determines the next execution. A branch step selects the next rule depending on the branch condition.

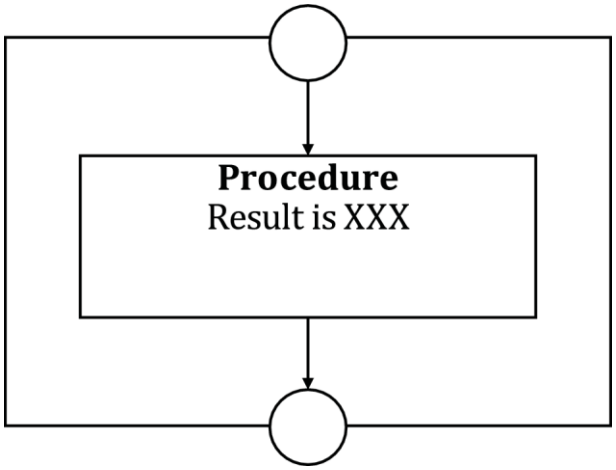

Figure 3 — Procedural step

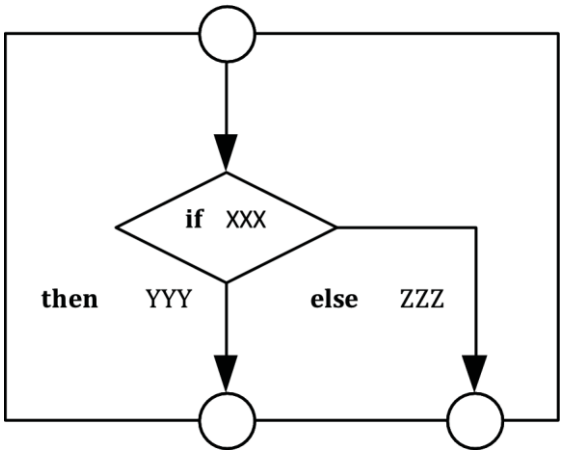

Figure 4 — Branch step

The rules shall define:

— the sequence of exchanging messages;

— the conditions of granting or denying access;

— the function-ID which specifies a request function for the Processing module and identifies the sender function of the Processing module in messages of the storage interface;

— the destination of Final-Result-Notification.

The rules should define:

— the destination and the timing of Time-stamp-Notification.

At least one rule is linked to Access ID.

7 Time stamping function

Each module except the Policy module has a time stamping function. The time stamping function is used to measure the duration of a transaction, request performance time and the processing time at each module. Usage of time stamping functions are shown in Annex C.

The time stamping function of each module records receivedTime and sendingTime in each response message. The time stamping function of the RED module also logs the time when it sends and receives messages.

8 Module

This clause describes the modules that are shown in the access system model (Figure 1).

8.1 Policy module

The Policy module is a module that defines the behaviour of an access system.

The Policy module shall keep the source of the rules.

The Policy module shall set the rules identified by rule-ID to the RED module.

8.2 Access-point module

The Access-point module is an interface module between an access system and Accessors.

The Access-point module receives an access request and generates a transaction.

When an Access-point module receives an Access-request including an access-ID, it shall generate a transaction-ID and Transaction-start-request and shall send it to the RED module.

The Access-point module shall have its own identifier as access-point-ID.

8.3 RED module

The RED module is a module for the rule evaluation and dispatching (RED) functions of a distributed real-time access system.

The RED module shall process a transaction and manage time stamping function (logging, notifications). These functions shall be controlled by the rules that are set by the Policy module.

To manage time stamping information, the RED module shall log receivedTime and sendingTime in each message. The RED module also shall log the time when it sends and receives messages. The RED module shall send Time-stamp-Notification to the receiver(s) through the external interface (Out).

8.4 Processing module

The Processing module is a module that executes various functions related to transactions.

The Processing module shall execute functions requested by the RED module.

When the Processing module receives a Processing-request from the RED module, it shall execute the function identified by function-ID in the Processing-request. After that it shall generate a Processing-response that includes the execution result and shall send it to the RED module.

The Processing module shall be able to send Store-request and Retrieve-request to the RED module for accessing data in the Storage module.

8.5 Storage module

The Storage module is a module that stores data related to transactions.

The Storage module shall store and retrieve data by requests from RED module.

When the Storage module receives a Store-request, the Storage module shall store the data, shall generate a Store-response and shall send it to the RED module. When the Storage module receives a Retrieve-request, the Storage module shall retrieve the data, shall generate a Retrieve-response that includes the retrieved data and shall send it to the RED module.

The Storage module may be used for sharing information between different transactions in the same access system or a different access system as shown in Annex B.

9 Messages of each interface

This clause specifies the messages which each module shall exchange via interfaces. Each message shall contain a number of elements specified in Clause 9. In this Standard, the messages are specified by an ASN.1 expression. Encoding rules are not specified.

Messages exchanged in the access system are shown in Annex D.1.

9.1 Messages of Policy interface

The Policy interface is the interface between the Policy module and the RED module. Policy-setter and Policy-getter messages are exchanged though the Policy interface.

The Policy module uses Policy-setter to set the rules for the RED module and may send Policy-setter at any time. The RED module may use Policy-getter to request the Policy module to set the rules at any time. Policy-getter is an optional message.

(1) Policy-setter

Policy-setter contains rule-ID and rule at least.

The structure of Policy-setter is as follows.

Policy-setter ::= SEQUENCE

```
{
    rule-ID             OCTET STRING,
    rule                OCTET STRING,
    …
}
```

(2) Policy-getter

Policy-getter contains rule-ID at least.

The structure of Policy-getter is as follows.

Policy-getter ::= SET

```
{
    rule-ID                 OCTET STRING,
    …
}
```

9.2 Message of Access interface

The Access interface is the interface between the Access point module and the RED module, and Transaction-start-request is sent through the access interface.

Transaction-start-request contains transaction-ID, access-point-ID and sendingTime at least.

The structure of Transaction-start-request is as follows.

Transaction-start-request ::= SET

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    sendingTime             GeneralizedTime,
    …
}
```

9.3 Messages of Processing interface

The processing interface is the interface between the RED module and the Processing module, and Processing-request, Processing-response, Store-request, Store-response, Retrieve-request, Retrieve-response are exchanged though the processing interface.

(1) Processing-request

Processing-request contains transaction-ID, function-ID and set-of-parameter at least.

The structure of Processing-request is as follows.

Processing-request ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            }
    function-ID             OCTET STRING,
    set-of-parameter        SET {
                            parameter                   OCTET STRING
                            }
    …
}
```

(2) Processing-response

Processing-response is the response message sent from the Processing module in response to a Processing-request sent from the RED module to the Processing Module.

Processing-response contains transaction-ID, function-ID, receivedTime, sendingTime and result at least.

The structure of Processing-response is as follows.

Processing-response ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            }
    function-ID             OCTET STRING,
    receivedTime            GeneralizedTime,
    sendingTime             GeneralizedTime,
    result                  OCTET STRING,
    …
}
```

receivedTime indicates the time at which the Processing module received the corresponding Processing-request from the RED module.

sendingTime indicates the time at which this response is sent.

result includes the result of executing the function.

(3) Store-request

Store-request is a request message for storing data sent from the Processing module to the Storage module through the RED module. Store-request contains Transaction ID, function-ID, data-type and data at least.

The structure of Store-request is as follows.

Store-request ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    data-type               OCTET STRING,
    data                    OCTET STRING
    …
}
```

(4) Retrieve-request

Retrieve-request is a message for retrieving data for execution of processing. It is sent from the Processing module to the Storage module through the RED module. Retrieve-request contains transaction-ID, function-ID and data-type at least.

The structure of Retrieve-request is as follows.

Retrieve-request ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    data-type               OCTET STRING,
    …
}
```

(5) Store-response

The RED module sends Store-response to the Processing module in response to a Store-request. Store-response contains transaction-ID, function-ID, receivedTime, sendingTime and result at least.

The structure of Store-response is as follows.

Store-response ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            }
    function-ID             OCTET STRING,
    receivedTime            GeneralizedTime,
    sendingTime             GeneralizedTime,
    result                  OCTET STRING,
    …
}
```

function-ID is the same as the function-ID in the corresponding Store-request.

receivedTime indicates the time at which the Storage module received the corresponding Store-request from the RED module.

sendingTime indicates the time at which this response is sent.

result indicates whether data in the corresponding Store-request is stored or not.

(6) Retrieve-response

The RED module sends Retrieve-response to the Processing module in response to Retrieve-request. Retrieve-request contains transaction-ID, function-ID, receivedTime, sendingTime and data at least.

The structure of Retrieve-response is as follows.

Retrieve-response ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    receivedTime            GeneralizedTime,
    sendingTime             GeneralizedTime,
    data                    OCTET STRING,
    …
}
```

function-ID contains the same data as function-ID in the corresponding Retrieve-request.

receivedTime indicates the time at which the Storage module received the corresponding Retrieve-request from the RED module.

sendingTime indicates the time at which this response is sent.

data includes the data which is retrieved upon the corresponding Retrieve-request.

9.4 Messages of Storage interface

The storage interface is the interface between the RED module and the Storage module. Store-request, Retrieve-request, Store-response and Retrieve-response messages are exchanged through the storage interface.

Store-request and Retrieve-request are sent from the RED module to the Storage module and Store-response and Retrieve-response are sent from the Storage module to the RED module in response to a request from the RED module

(1) Store-request

Store-request contains transaction-ID, function-ID data-type and data at least.

The structure of Store-request is as follows.

Store-request ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    data-type               OCTET STRING,
    data                    OCTET STRING,
    …
}
```

(2) Retrieve-request

Retrieve-request contains transaction-ID, function-ID and data-type at least.

The structure of Retrieve-request is as follows.

Retrieve-request ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    data-type               OCTET STRING,
    …
}
```

(3) Store-response

The Storage module sends Store-response to the RED module in response to Store-request. Store-response contains transaction-ID, function-ID, receivedTime, sendingTime and result at least.

The structure of Store-response is as follows.

Store-response ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING ,
    receivedTime            GeneralizedTime,
    sendingTime             GeneralizedTime,
    result                  OCTET STRING,
    …
}
```

(4) Retrieve-response

The Store module sends Retrieve-response to the RED module in response to Retrieve-request. Retrieve-response contains transaction-ID, function-ID, receivedTime, sendingTime and data at least.

The structure of Retrieve-response is as follows.

Retrieve-response ::= SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    function-ID             OCTET STRING,
    receivedTime            GeneralizedTime,
    sendingTime             GeneralizedTime,
    Data                    OCTET STRING,
    …
}
```

10 Messages of external interfaces

This clause specifies an access request and notifications between an access system and entities located outside of the access system as shown in Figure 1. Each message shall contain a number of elements specified in Clause 10. In this Standard, the messages are specified by an ASN.1 expression. Encoding rules are not specified.

Messages exchanged between inside and outside of the access system are shown in Annex D.2.

10.1 Access request from external interface (In)

Access-request is a request sent by an Accessor and is received by the Access point module.

Access-request contains access-ID at least, and its structure is as follows.

Access-request ::= SET

```
{
    access-ID           OCTET STRING,
    …
}
```

10.2 Final result notification to external interface (Out)

Final-Result-Notification is a notification of the final result of a transaction (grant or deny). It is generated by the RED module using the messages such as Processing-response and Store-response from the Processing module and the Storage module according to the rule.

Final-Result-Notification contains transaction-ID, resultOfTransaction at least, and its structure is as follows.

Final-Result-Notification :: = SEQUENCE

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    resultOfTransaction     ENUMERATED{ GRANT, DENY },
    …
}
```

10.3 Time stamp notification

Time-stamp-Notification is a notification to output time stamp information.

The RED module sends Time-stamp-Notification to the receiver(s). Time-stamp-Notification contains transaction-ID, Time-Stamp-Information at least and its structure is as follows.

Time-stamp-Notification :: = SETOF

```
{
    transaction-ID          SEQUENCE {
                            access-ID                   OCTET STRING,
                            access-point-ID             OCTET STRING,
                            access-ID-obtained-time     GeneralizedTime,
                            …
                            },
    Time-Stamp-Information  CHOICE {
                                transactionProcessingTime               GeneralizedTime,
                                requestPerformanceTimes                 SET OF GeneralizedTime,
                                moduleProcessingTimes                   SET OF GeneralizedTime,
                                dataTransmissionTime                    SET OF GeneralizedTime,
                                requestPerformanceTimesForRetrieve      SET OF GeneralizedTime,
                                moduleProcessingTimesForRetrieve        SET OF GeneralizedTime,
                                dataTransmissionTimeForRetrieve         SET OF GeneralizedTime,
                                requestPerformanceTimesForStore         SET OF GeneralizedTime,
                                moduleProcessingTimesForStore           SET OF GeneralizedTime,
                                dataTransmissionTimeForStore            SET OF GeneralizedTime,
                                },
    …
}
```

transactionProcessingTime, Request processing, Module processing time and dataTransmissionTime are calculated at RED module (see 11.1).

11 Access system performance management

A performance management scheme is introduced to confirm the conformance of the real-time performance of an access system.

Performance management mechanisms allow an access system to be evaluated for performance by using specific elements and metrics, such as maximum time, average time and standard deviation. Those metrics are measured by logged time in the RED module.

The RED module can measure the following time:

1) transaction processing time;

2) request performance time;

3) module processing time;

4) data transmission time.

The choice of performance elements and metrics depends on application and service requirements. For example, the performance is specified “The maximum time (metric) of transactionProcessingTime (element) is less than x seconds (value)”.

11.1 Transaction processing time

transactionProcessingTime (t_{tp}) shown in Figure 5 describes the performance of an access system to process a transaction. t_{tp} is measured by calculating the difference between the following two values for a transaction at the RED module:

(1) t_{R1} : The time when receiving Transaction-start-request from the Access-point module.

(2) t_{R2} : The time when sending Final-Result-Notification corresponding to the Transaction-start-request in (1).

$$t_{tp} = t_{R2} - t_{R1} \quad (11.1)$$

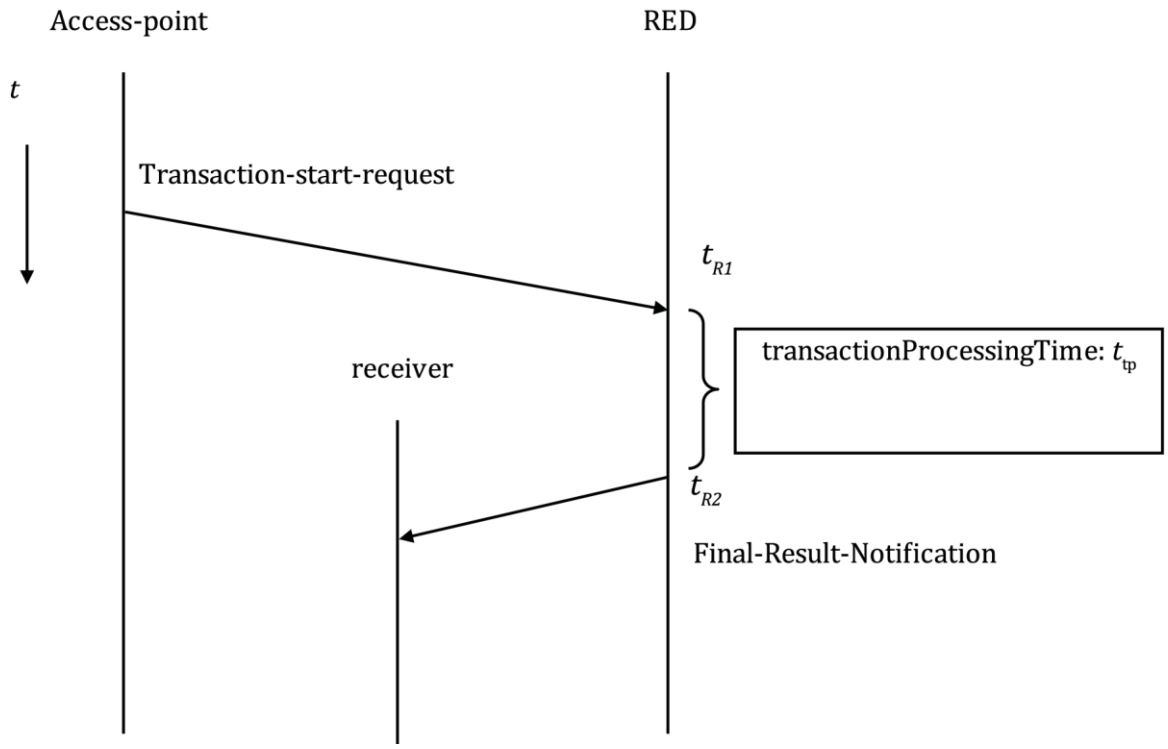

Figure 5 — transactionProcessingTime

11.2 Request performance time

Request performance time (t_{rp}) shown in Figure 6 describes the performance of the Processing module and the data transmission time. t_{rp} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{R3} : The time when sending a Processing Request.

(2) t_{R4} : The time when receiving a Processing Response corresponding to the Processing Request in (1).

$$t_{rp} = t_{R4} - t_{R3} \tag{11.2}$$

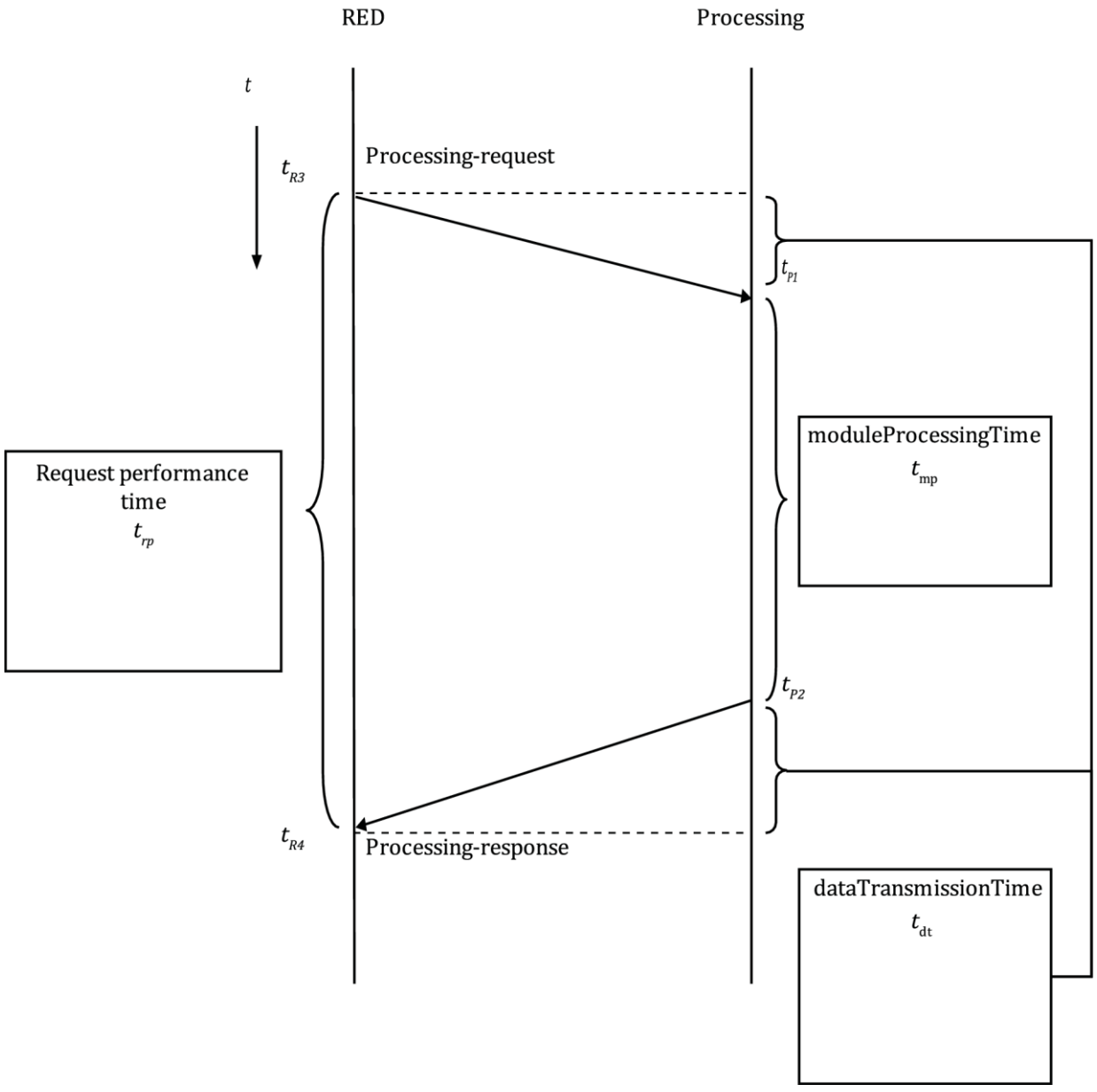

Figure 6 — Request performance time, module processing time and dataTransmissionTime

11.3 Module processing time

Module processing time (t_{mp}) shown in Figure 6 describes the performance of the Processing module. The data transmission time is not included in t_{mp}. t_{mp} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{P1} : receivedTime stored in a Processing-response.

(2) t_{P2} : sendingTime stored in the same Processing-response with (1).

$$t_{mp} = t_{P2} - t_{P1} \tag{11.3}$$

11.4 Data transmission time

dataTransmissionTime (t_{dt}) shown in Figure 6 describes the performance of the network which connects the RED module and the Processing module. t_{dt} is measured by calculating the difference between t_{rp} and t_{mp}.

$$t_{dt} = t_{rp} - t_{mp} \tag{11.4}$$

11.5 Request performance time for retrieve

Request performance time for retrieve (t_{rpr}) shown in Figure 7 describes the performance of retrieving data from the Storage module. t_{rpr} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{R5} : The time when sending a Retrieve request.

(2) t_{R6} : The time when receiving a Retrieve response corresponding to the Retrieve request in (1).

$$t_{rpr} = t_{R6} - t_{R5} \tag{11.5}$$

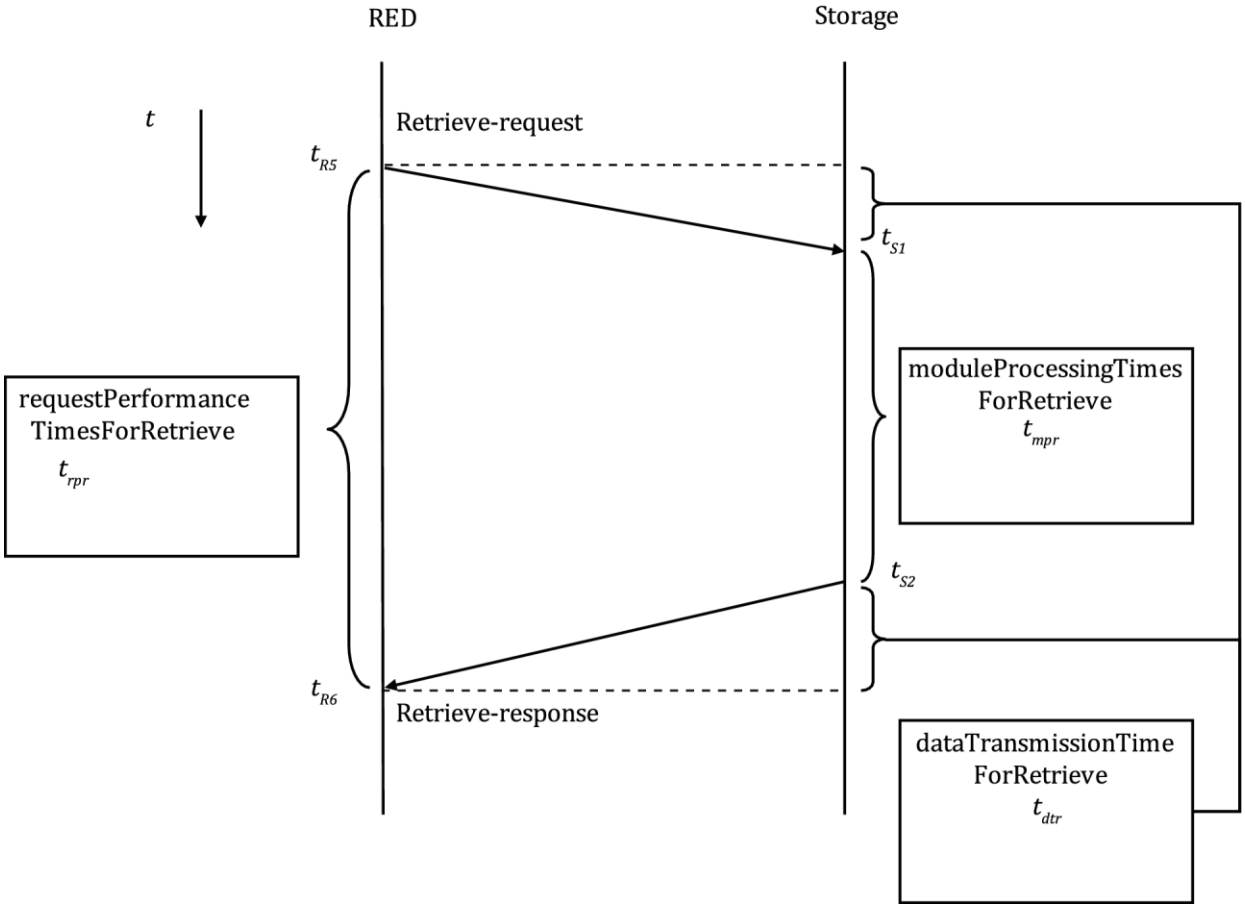

Figure 7 — Request performance time for retrieve, module processing time for retrieve and dataTransmissionTimeForRetrieve

11.6 Module processing time for retrieve

Module processing time for retrieve (t_{mpr}) describes the performance of retrieving data in the Storage module. The data transmission time is not included in t_{mpr}. t_{mpr} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{S1} : receivedTime stored in a Retrieve-response.

(2) t_{S2} : sendingTime stored in the same Retrieve-response with (1).

$$t_{mpr} = t_{S2} - t_{S1} \quad (11.6)$$

11.7 Data transmission time for retrieve

dataTransmissionTimeForRetrieve (t_{dtr}) describes the performance of the network which connects the RED and Storage modules. t_{dtr} is measured by calculating the difference between t_{rpr} and t_{mpr}.

$$t_{dtr} = t_{rpr} - t_{mpr} \quad (11.7)$$

11.8 Request performance time for store

Request performance time for store (t_{rps}) describes the performance of storing data from storage module as shown in Figure 8. t_{rps} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{R7} : The time when sending a Store request.

(2) t_{R8} : The time when receiving a Store response corresponding to the Store request in (1).

$$t_{rps} = t_{R8} - t_{R7} \quad (11.8)$$

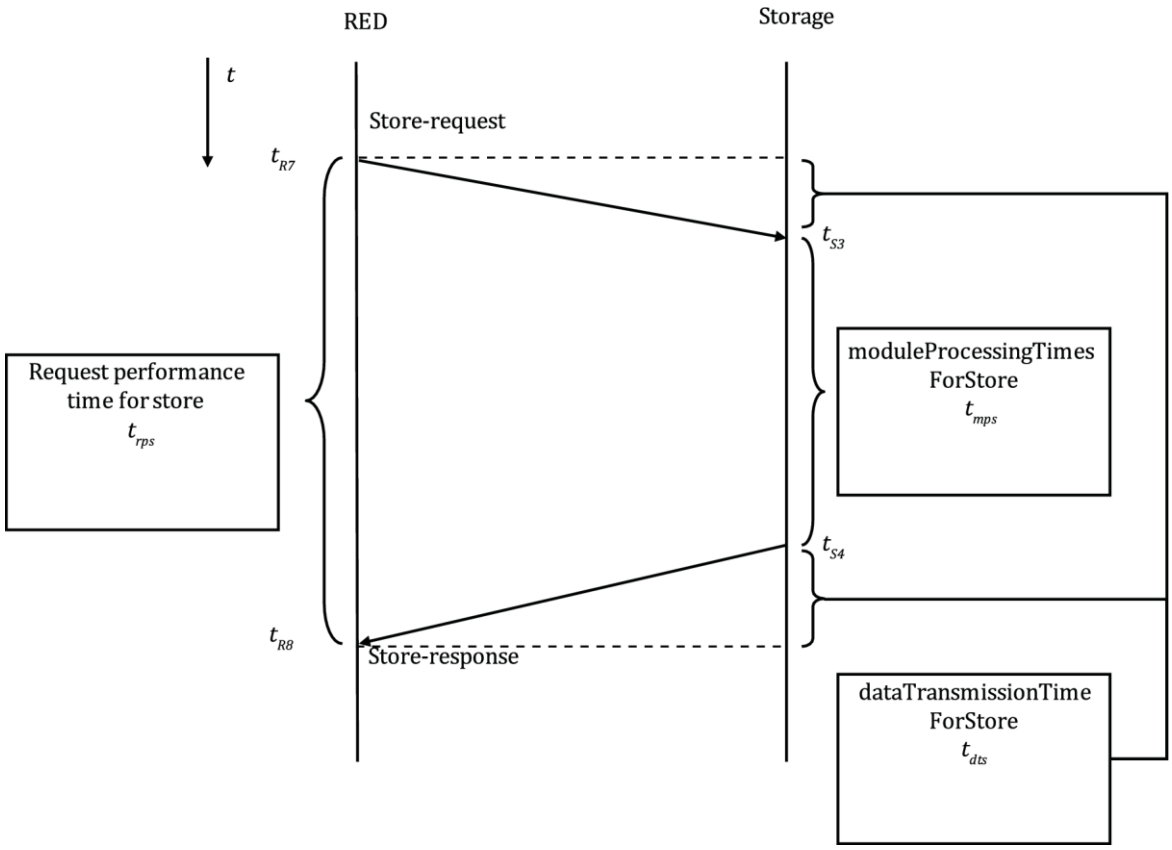

Figure 8 — Request performance time for store, module processing time for store and dataTransmissionTimeForStore

11.9 Module processing time for store

Module processing time for store (t_{mps}) describes the performance of storing data in the storage module. The data transmission time is not included in Module processing time for store. t_{mps} is measured by calculating the difference between the following two values for a set of Request message and Response message logged by the RED module:

(1) t_{S3} : receivedTime stored in a Store-response.

(2) t_{S4} : sendingTime stored in the same Store-response with (1).

$$t_{mps} = t_{S4} - t_{S3} \quad (11.9)$$

11.10 Data transmission time for store

dataTransmissionTimeForStore (t_{dts}) describes the performance of the network which connects the RED and Storage modules. t_{dts} is measured by calculating the difference between t_{rps} and t_{mps}. t_{dts} are dataTransmissionTimeForStore_request and dataTransmissionTimeForStore_response:

$$t_{dts} = t_{rps} - t_{mps} \quad (11.10)$$

11.11 Access point processing time

Access point processing time (t_{ap}) shown in Figure 9 describes the performance of the access point module. The data transmission time is not included in t_{ap}. t_{ap} is measured by calculating the difference between the following two values for a set of Request message logged by the RED module:

(1) t_{A1} : access-ID-obtained-time stored in a Transaction-start-request.

(2) t_{A2} : sendingTime stored in the same Transaction-start-request with (1).

$$t_{ap} = t_{A2} - t_{A1} \tag{11.11}$$

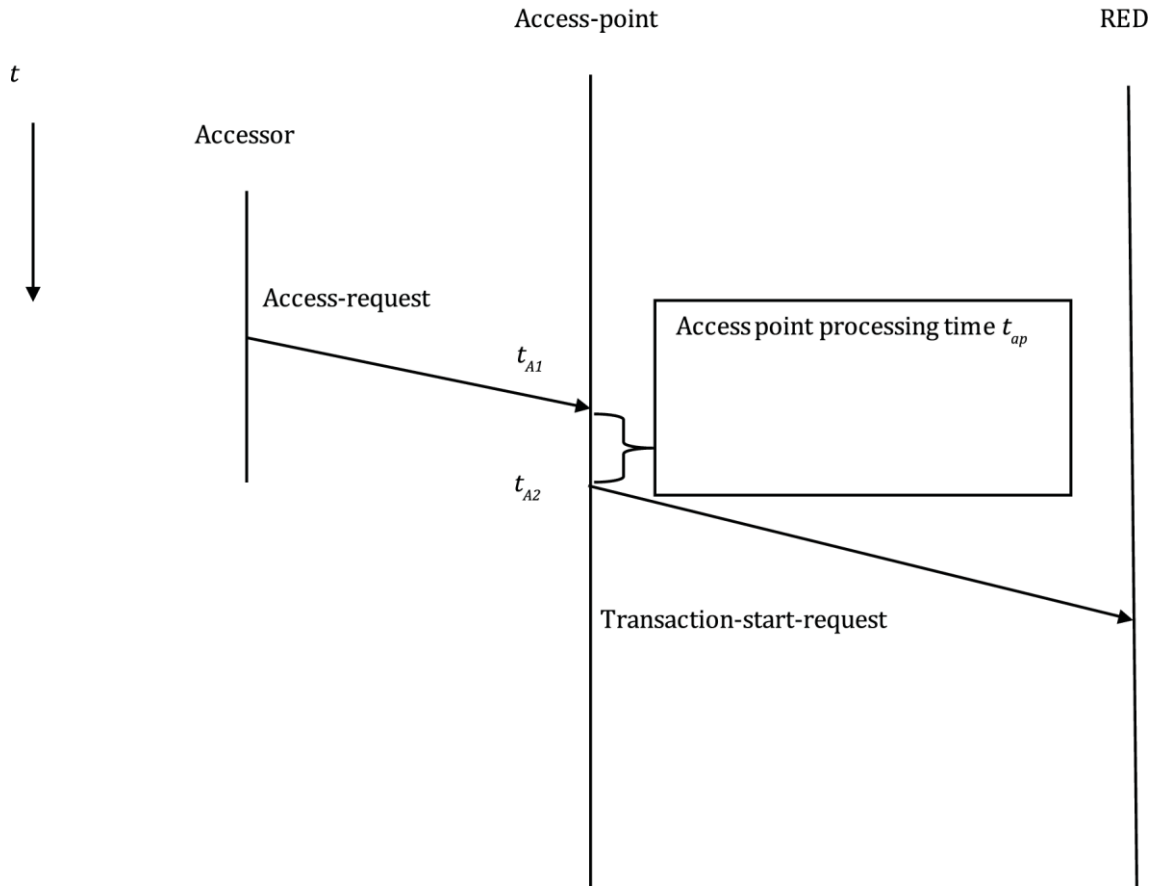

Figure 9 — Access point processing time

Annex A
(informative)

Service access control system

This service access control system provides an authentication process for ensuring that the user can use the device for access request.

An example of message sequence for the validation process is shown in Figure A.1.

The Policy module sends Policy-setter to the RED module to set the rules.

When the Access-point module obtains access-ID from a device which user access the Access Point module, then it sends a Transaction-start-request to the RED module.

The RED module interprets the rules based on the Transaction-start-request, and then it sends a Processing-request with function-ID for validation to the Processing module to execute a validation function to validate access-ID.

The Processing module executes a validation function and then it sends a Processing-response including the result (OK or NG) to the RED module.

The RED module sends a Final-Result-Notification including the final result (grant or deny) to the receiver.

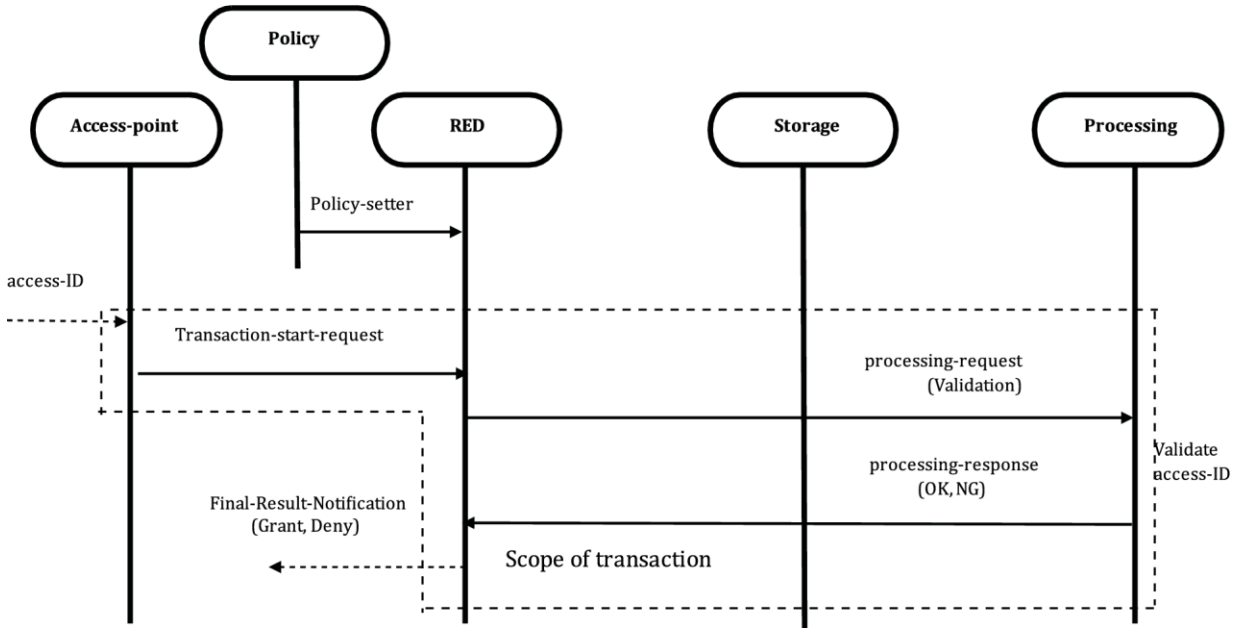

Figure A.1 — An example of message sequence for the authentication process

Annex B
(informative)

Share information between different access systems

This is another access system use case which allow to use the same User ID for other access system.

Example of the use case is the following;

User uses an access device at Facility X. The access device has User ID and can be used as usage log store device, Storage module. The usage log at Facility X is then stored in a Storage module (see Figure B.1 <1>). After using it at Facility X, user uses it at Facility Y with the same ID. In order to provide a combination service between Facility X and Y (e.g. discount, point program and so on), the usage log which is stored in Storage module as shared information for both Facility X and Facility Y is used by Facility Y (see Figure B.1 <2>).

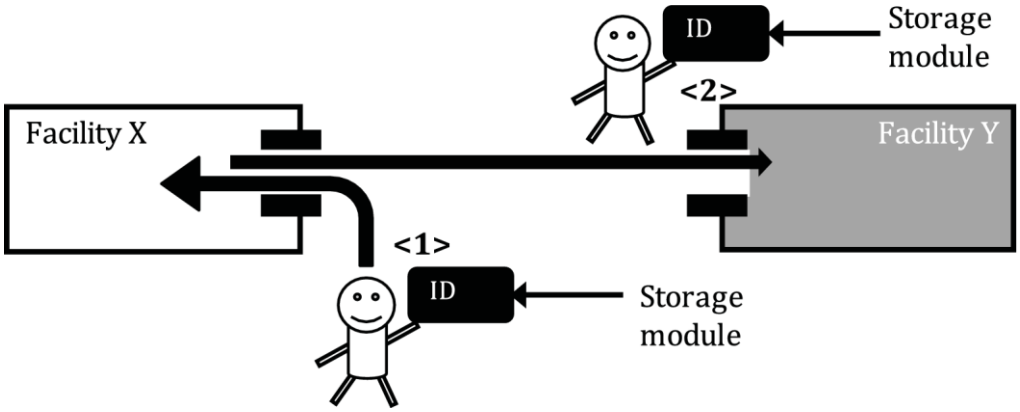

Figure B.1 — An example of shared information application

For these purposes, access systems may have a Storage module which is shared with another access system.

The example of two access systems which have shared a Storage module is as follows:

In order to achieve the information shared access system, one Storage module will be connected to another access system and can be used at both access systems (see Figure B.2).

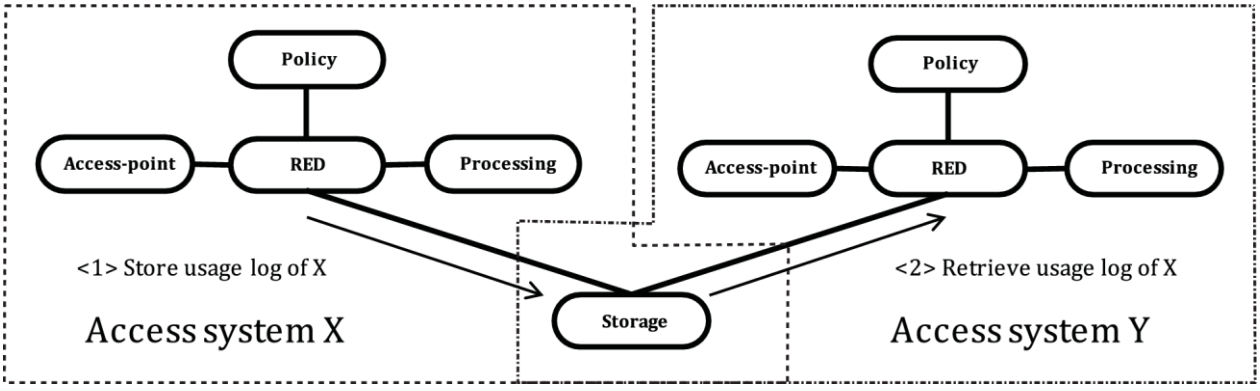

Figure B.2 — An example of Storage module shared access system

Annex C
(informative)

Usage of time stamping

This Annex C shows the time stamping data stored timing to RED module, Processing module and Storage module in the typical example of message sequences (see Figure C.1).

The RED module is able to measure the following duration times:

A) **Access point processing time (t_{ap})** = $t_{A2} - t_{A1}$

B) **transactionProcessingTime (t_{tp})** = $t_{R14} - t_{R3}$

C) **Request performance time (t_{rp})** = $t_{R11} - t_{R2}$

D) **Module processing time (t_{mp}) =** $t_{P6} - t_{P1}$

E) **dataTransmissionTime (t_{dt})** = $t_{rp} - t_{mp}$

F) **Request performance time for retrieve (t_{rpr})** = $t_{R7} - t_{R6}$

G) **Module processing time for retrieve (t_{mpr})** = $t_{S2} - t_{S1}$

H) **dataTransmissionTimeForRetrieve (t_{dtr}) =** $t_{rpr} - t_{mpr}$

I) **Request performance time for store (t_{rps})** = $t_{R11} - t_{R10}$

J) **Module processing time for store (t_{mps})** = $t_{S4} - t_{S3}$

K) **dataTransmissionTimeForStore (t_{dts}) =** $t_{rps} - t_{mps}$

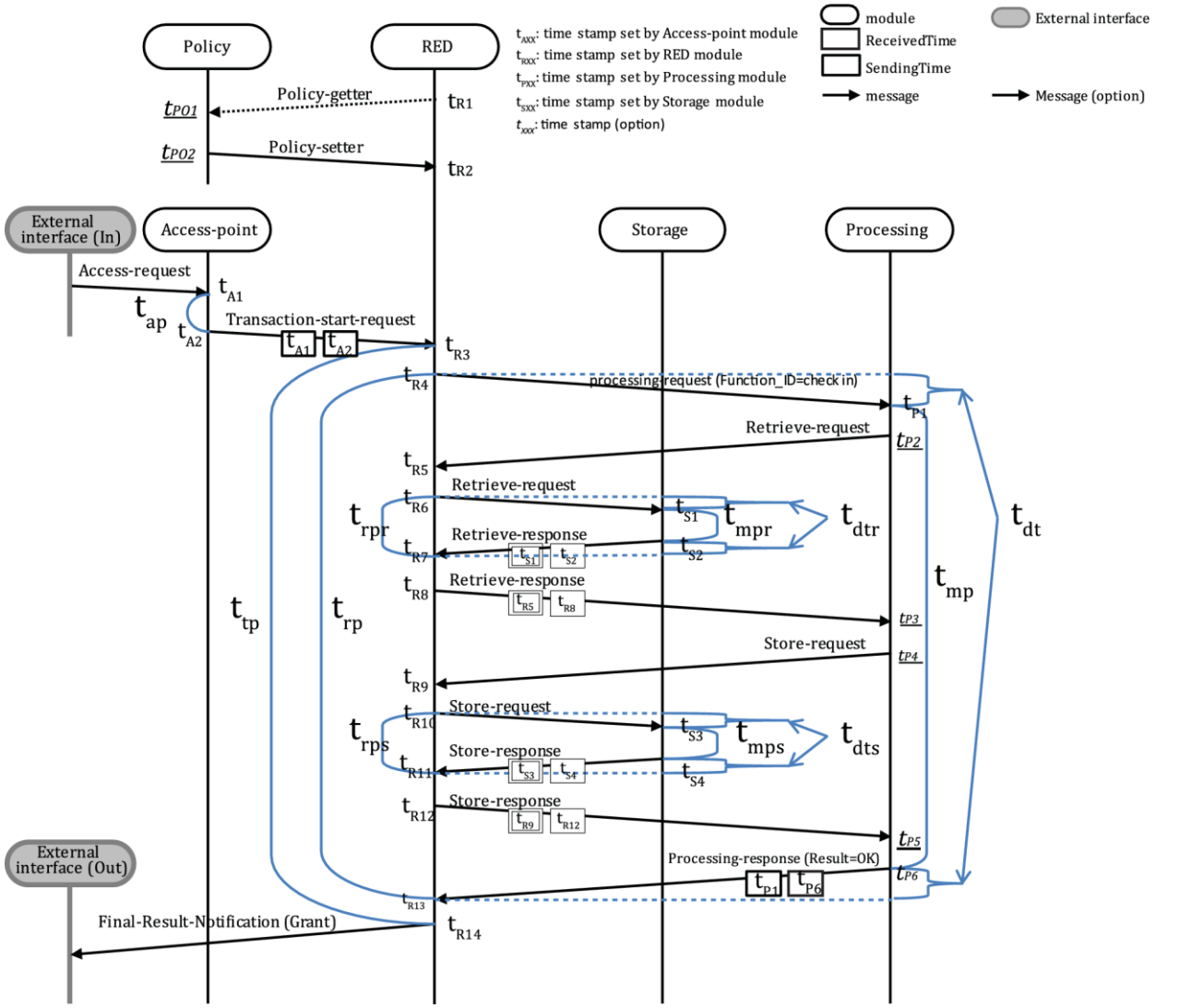

Figure C.1 — An example of time stamping and time measurement

(1) RED module sends Policy-getter at t_{R1}.

(2) Policy module receives Policy-getter at t_{PO1}.

(3) Policy module sends Policy-setter at t_{PO2}.

(4) RED module receives Policy-getter at t_{R2}.

(5) Access point module receives Access ID at t_{A1}.

(6) Access point module sends Transaction-start-request at t_{A2}.

(7) RED module receives Transaction-start-request at t_{R3}.

(8) RED module sends Processing-request at t_{R4}.

(9) Processing module receives Processing-request at t_{P1}.

(10) RED module receives Retrieve-request at t_{R5}.

(11) RED module sends Retrieve-request at t_{R6}.

(12) Storage module receives Retrieve-request at t_{S1}.

(13) Storage module sends Retrieve-response at t_{S2}.

(14) RED module receives Retrieve-response at t_{R7}.

(15) RED module sends Retrieve-response at t_{R8}.

(16) RED module receives Store-request at t_{R9}.

(17) RED module sends Store-request at t_{R10}.

(18) Storage module receives Store-request at t_{S3}.

(19) Storage module sends Store-response at t_{S4}.

(20) RED module receives Store-response at t_{R11}.

(21) RED module sends Store-response at t_{R12}.

(22) Processing module sends Processing-response at t_{P6}.

(23) RED module receives Processing-response at t_{R13}.

(24) RED module sends Final-Result-Notification at t_{R14}.

Annex D
(informative)

List of messages

D.1 Messages of each interface

No	Message	Interface	Direction		Semantics
			from	to	
1	Policy-setter	Policy Interface	Policy	RED	Set the rules to the RED module
2	Policy-getter		RED	Policy	Request to set rules to the RED module
3	Transaction-start-request	Access Interface	Access-point	RED	Request to start a new transaction
4	Processing-request	Processing Interface	RED	Processing	Request to process a function
5	Processing-response		Processing	RED	Result of processing
6	Store-request		Processing	RED	Request to store data
7	Store-response		RED	Processing	Result of storing
8	Retrieve-request		Processing	RED	Request to retrieve data
9	Retrieve-response		RED	Processing	Retrieved data
10	Store-request	Storage Interface	RED	Storage	Request to store data
11	Store-response		Storage	RED	Result of storing
12	Retrieve-request		RED	Storage	Request to retrieve data
13	Retrieve-response		Storage	RED	Retrieved data

D.2 Messages of external interface

No	Message	Direction		Semantics
		from	to	
1	Access-request	Accessor	Access-point	Request trigger of processing for access system
2	Final-Result-Notification	RED	Receiver	Notification of the final result of a transaction (grant or deny)
3	Time-stamp-Notification	RED	Receiver	Notification to output time stamp information