Standard ECMA-412

2nd Edition / June 2017

Framework for distributed real-time Access systems

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.

This 2nd edition of the Standard introduces 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 Ecma Standard was developed by Technical Committee 51 and was adopted by the General Assembly of June 2017.

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 different 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 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.

3 Normative references

None.

4 Terms, definitions and acronyms

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

4.1
Accessor
Someone or something that interacts with the Access system

4.2
Access_ID
the Identifier in an Access request

4.3
Access_ID_obtained_time
the time when an Access point module obtains an Access_ID

4.4
Access-point_ID
the identifier of an Access-point module

4.5
Access_request
a request trigger of processing for access system

4.6
Final_Result_Notification
a notification of the final result of a transaction

4.7
Function_ID
the identifier of function

4.8
Policy_getter
a message to request the Policy module to set the rules

4.9
Policy_setter
a message to set the rules to the RED module

4.10
Processing_request
a request to execute a function

4.11
Processing_response
a response to a Processing_request

4.12
RED
Rule Evaluation and Dispatching

4.13
ReceivedTime
the time when a module receives a request from another module

4.14
Retrieve_request
a request to retrieve data from storage

4.15
Retrieve_response
a response to a Retrieve_request

4.16
Rule_ID
the identifier of rules

4.17
SendingTime
the time when a module sends a response or a Transaction_start_request to another module

4.18
Store_request
a request to store data to storage

4.19
Store_response
a response to a Store_request

4.20
TimeStampingFlag
a request for the activation or deactivation of the time stamping function in a module

4.21
Time_stamp_Notification
a notification to provide time stamp information

4.22
Transaction_ID
the identifier of a transaction

4.23
Transaction_start_request
a request to initiate a transaction

5 Overview

This clause is an overview of the system model 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.

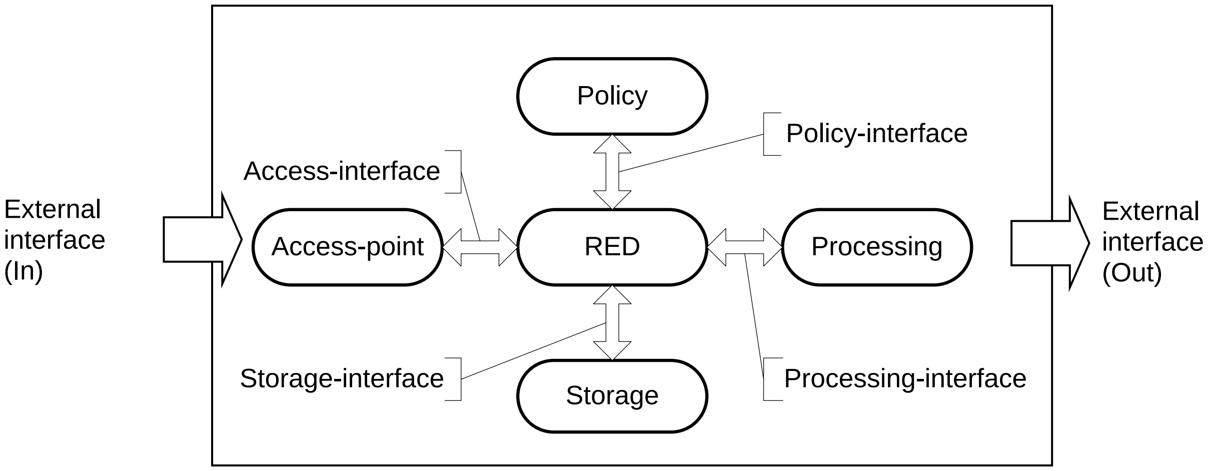

Figure 1 — Access system model

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 and 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.

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

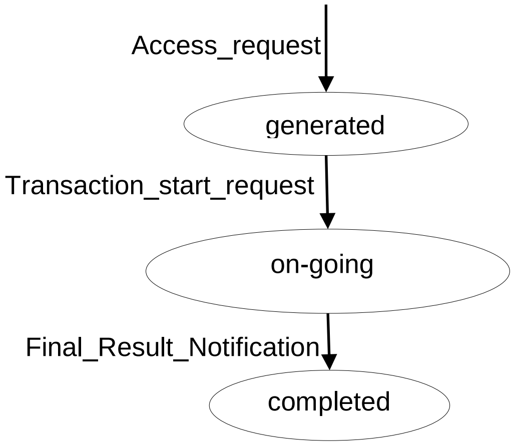

Figure 2 — Transaction State Machine

7 Time stamping function

The time stamping function which shall be provided by each module, except the Policy module, is used to measure the duration of a transaction, request performance time and the processing time at each module.

The time stamping function of an Access-point module is always activated. For the other modules, time stamping functions are activated and deactivated by controlling the Time_Stamping_Flag value in the requests from the RED module. The Time_Stamping_Flag value is set according to the rule.

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

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
- the destination of Final_Result_Notification
- the activation and the de-activation of time stamp functions by setting the value of Time_Stamp_Flag.

The rules should define:

- the destination and the timing of Time_Stamp_Notification.

At least one rule is linked to Access ID.

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

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 as long as the Time stamping function is activated. The RED module shall send Time_stamp_Notification to the receiver(s) through the external interface (Out).

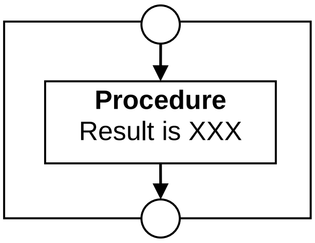

Figure 3 — Procedure rule

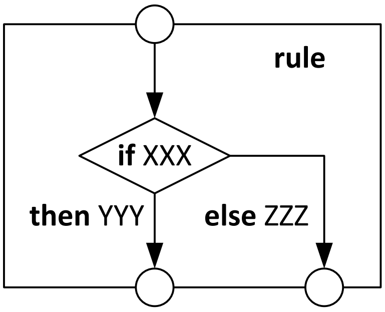

Figure 4 — Branch rule

8.4 Processing module

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 shall store and retrieve data related to transactions.

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 document, the messages are specified by an ASN.1 expression. Encoding rules are not specified.

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 and its structure is as follows.

```
Policy_setter ::= SEQUENCE {
      RULE_ID                   OCTET STRING,
      RULE                      OCTET STRING,
      …
}
```

(2) Policy_getter

Policy_getter contains RULE_ID at least and its structure 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 Transastion_start_request is sent through the access interface.

Transaction_start_request contains Transaction_ID at least and its structure 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, Time_Stamping_Flag and Set_Of_Parameter at least and its structure 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,
      Time_Stamping_Flag        BOOLEAN,
      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, Result at least. Processing_response contains ReceivedTime and SendingTime if Time_Stamping_Flag in the corresponding Processing_request is set to active.

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 Proccessing module to the Storage module through the RED module. Store_request contains Transaction ID, Function_ID, Time_Stamping_Flag, Data_type, Data at least and its structure 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,
      Time_Stamping_Flag          BOOLEAN,
      Data_type                   OCTET STRING,
      Data                        OCTET STRING
      …
}
```

Time_Stamping_Flag is the same as the Time_Stamping_Flag in the Processing_request that has the same Transaction_ID.

(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, Time_Stamping_Flag, Data_type at least and its structure 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,
      Time_Stamping_Flag          BOOLEAN,
      Data_type                   OCTET STRING,
      …
}
```

Time_Stamping_Flag is the same as the Time_Stamping_Flag in the Processing_request that has the same Transaction_ID.

(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, Result at least. Store_response contains ReceivedTime and SendingTime if the Time_Stamping_Flag in the corresponding Store_request is activated.

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 toRetrieve_request. Retrieve_request contains Transaction_ID, Function_ID, Data at least. Retrieve_response contains ReceivedTime and SendingTime if Time_Stamping_Flag in the corresponding Retrieve_request is set to active.

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, Time_Stamping_Flag, Data_type, Data at least and its structure 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,
    Time_Stamping_Flag      BOOLEAN,
    Data_type               OCTET STRING,
    Data                    OCTET STRING,
    …
}
```

(2) Retrieve_request

Retrieve_request contains Transaction_ID, Function_ID, Time_Stamping_Flag, Data_type at least and its structure 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,
    Time_Stamping_Flag      BOOLEAN,
    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, Result at least. Store_response contains ReceivedTime and SendingTime if the Time_Stamping_Flag in the corresponding Store_request is set to active.

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, Data at least. Retrieve_response contains ReceivedTime and SendingTime if the Time_Stamping_Flag in the corresponding Retrieve_request is set to active.

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 document, the messages are specified by an ASN.1 expression. Encoding rules are not specified.

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, Result_Of_Transaction 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). 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{
                                      Transaction processing time              Generalized time,
                                      Request performance times                SET OF GeneralizedTime,
                                      Module processing times                  SET OF GeneralizedTime,
                                      Data transmission time                   SET OF GeneralizedTime,
                                      Request performance times for retrieve   SET OF GeneralizedTime,
                                      Module processing times for retrieve     SET OF GeneralizedTime,
                                      Data transmission time for retrieve      SET OF GeneralizedTime,
                                      Request performance times for store      SET OF GeneralizedTime,
                                      Module processing times for store        SET OF GeneralizedTime,
                                      Data transmission time for store         SET OF GeneralizedTime,
                                      },
      …
}
```

Transaction processing time, Request processing, Module processing time and Data transmission time are calculated at RED module (See 11.1).

11 Access system performance management

A performance management scheme is introduced to confirm the conformance of performance requirements for 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 Transaction processing time (element) is less than x seconds (value)”.

11.1 Transaction processing time

Transaction processing time (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)

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

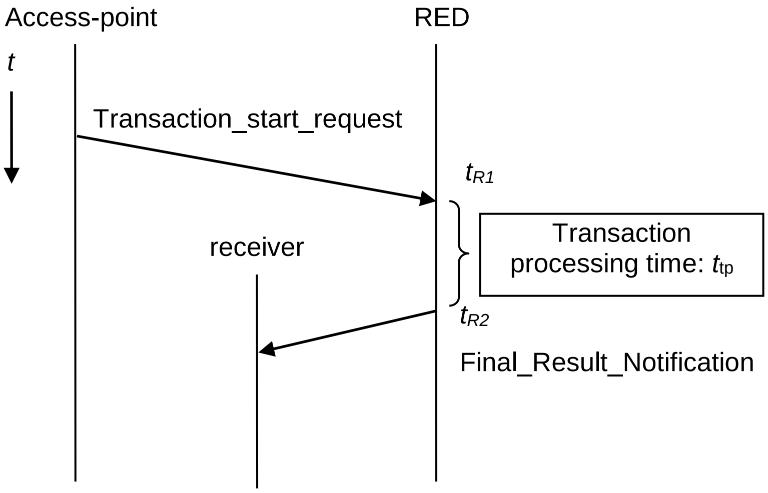
Access-point
RED
t
Transaction_start_request
t_{R1}
receiver
Transaction processing time: t_{tp}
t_{R2}
Final_Result_Notification

Figure 5 — Transaction processing time

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)

$$t_{rp} = t_{R4} - t_{R3} \qquad (11.2)$$

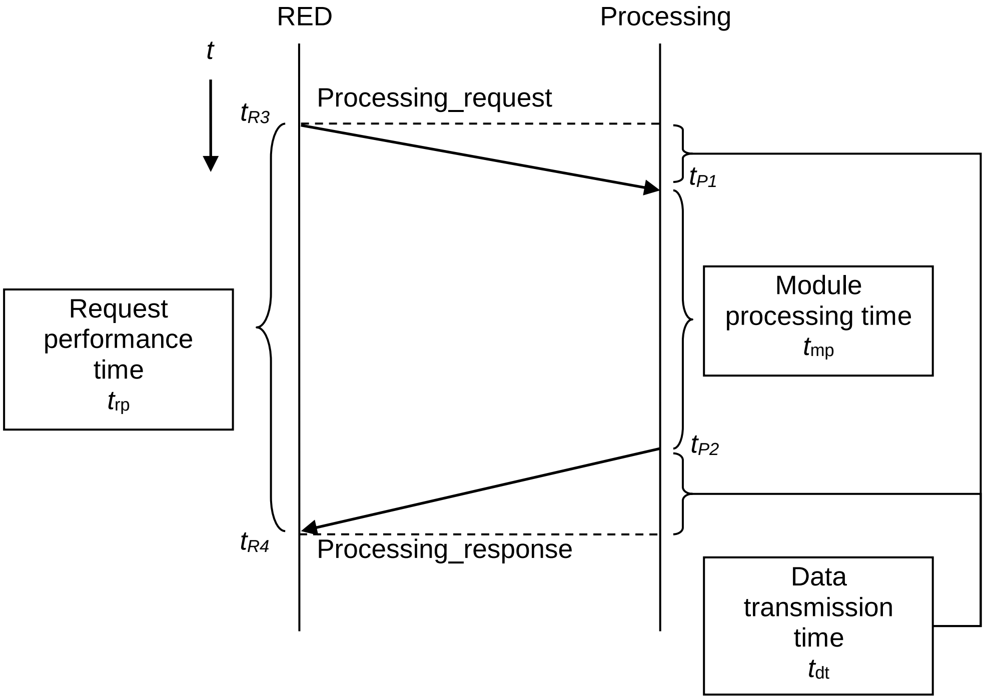
RED
Processing
t
Processing_request
t_{R3}
t_{P1}
Request performance time t_{rp}
Module processing time t_{mp}
t_{P2}
t_{R4}
Processing_response
Data transmission time t_{dt}

Figure 6 — Request performance time, Module processing time and Data transmission time

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)

$t_{mp} = t_{P2} - t_{P1}$ *(11.3)*

11.4 Data transmission time

Data transmission time (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}.

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

$t_{\mathrm{rpr}} = t_{R6} - t_{R5}$ *(11.5)*

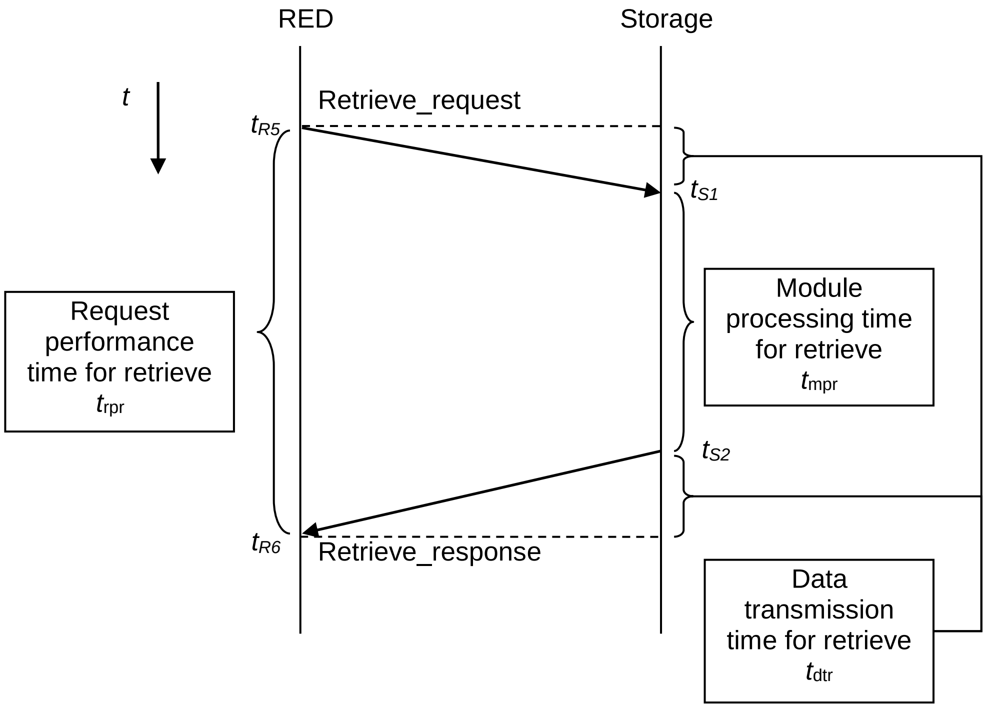
RED
Storage
t
Retrieve_request
t_{R5}
t_{S1}
Request performance time for retrieve t_{rpr}
Module processing time for retrieve t_{mpr}
t_{S2}
t_{R6}
Retrieve_response
Data transmission time for retrieve t_{dtr}

Figure 7 — Request performance time for retrieve, Module processing time for retrieve and Data transmission time for retrieve

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)

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

11.7 Data transmission time for retrieve

Data transmission time for retrieve (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} .

$t_{\mathrm{dtr}} = t_{\mathrm{rpr}} - t_{\mathrm{mpr}}$ *(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. 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)

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

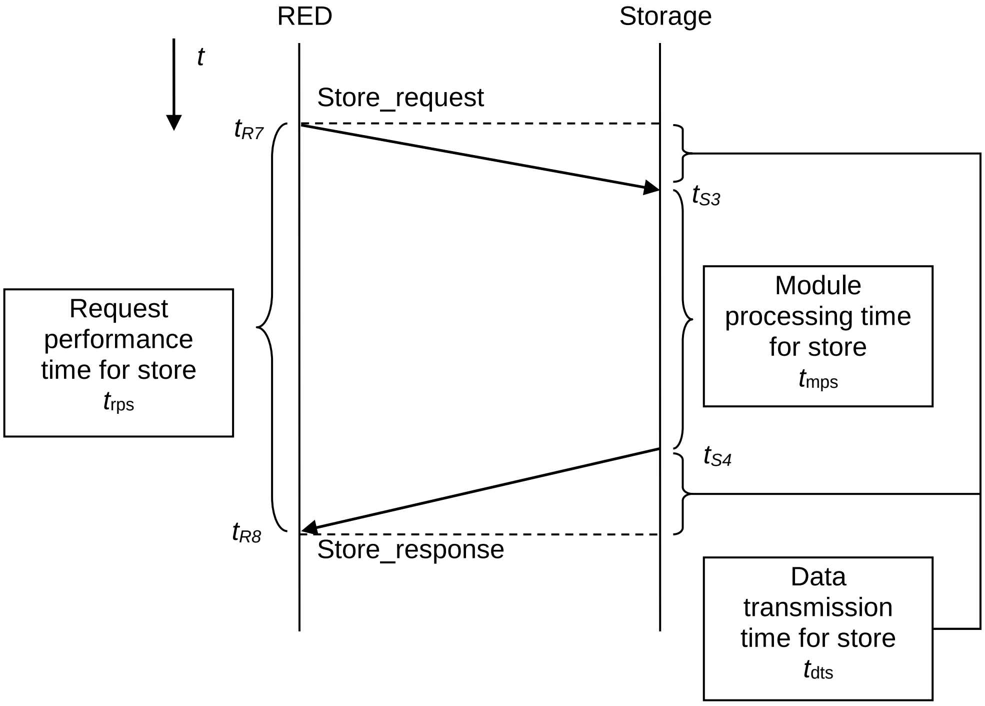
RED
Storage
t
Store_request
t_{R7}
t_{S3}
Module processing time for store t_{mps}
Request performance time for store t_{rps}
t_{S4}
t_{R8}
Store_response
Data transmission time for store t_{dts}

Figure 8 — Request performance time for store, Module processing time for store and Data transmission time for store

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)

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

11.10 Data transmission time for store

Data transmission time for store (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 Data transmission time for store_request and Data transmission time for store_response:

$t_{\mathrm{dts}} = t_{rps} - t_{mps}$ *(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)

$t_{ap} = t_{A2} - t_{A1}$ *(11.11)*

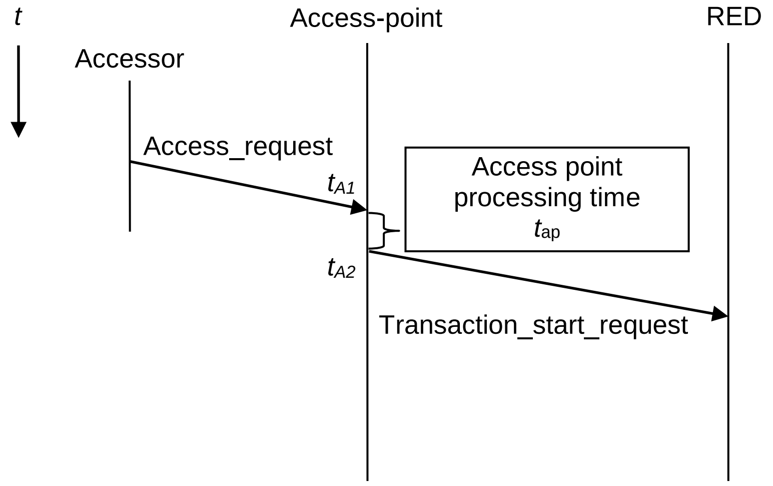
t
Access-point
RED
Accessor
Access_request
t_{A1}
Access point processing time t_{ap}
t_{A2}
Transaction_start_request

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 (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.

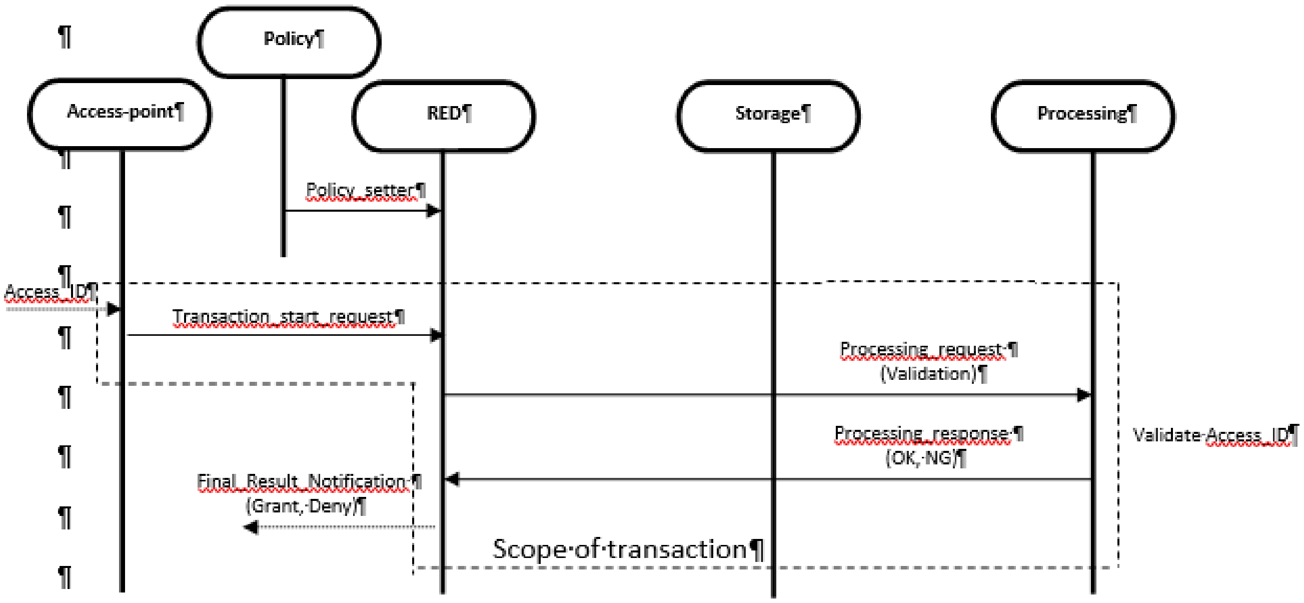

Figure A.1 — An example of message sequence for the authentication process

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>)

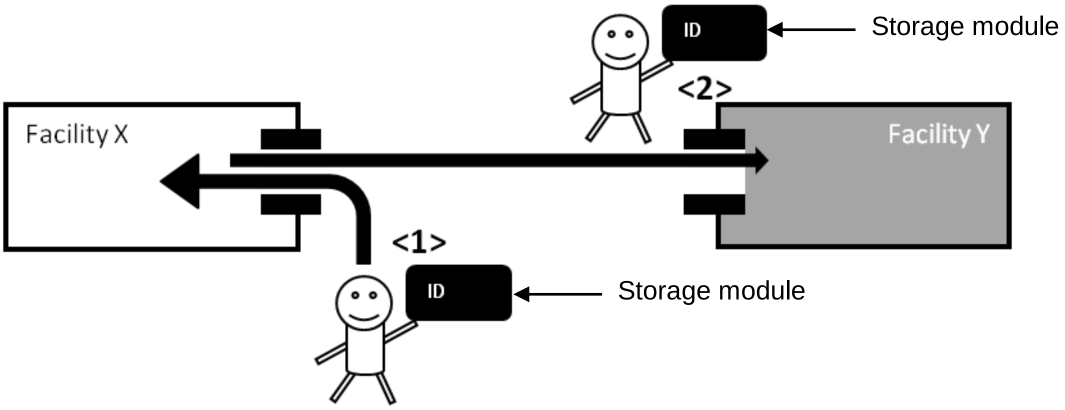

Figure B.1 — An example of shared information application

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)

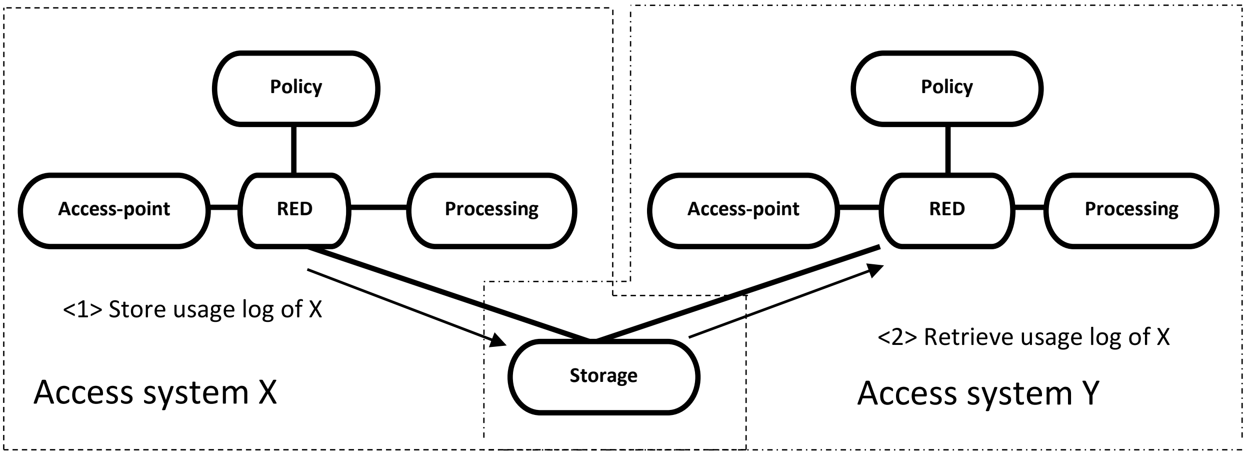

Figure B.2 — An example of Storage Module shared Access system

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) **Transaction processing time**（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) **Data transmission time** （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) **Data transmission time for retrieve**（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) **Data transmission time for store**（t_{dts}） = t_{rps} **-** t_{mps}

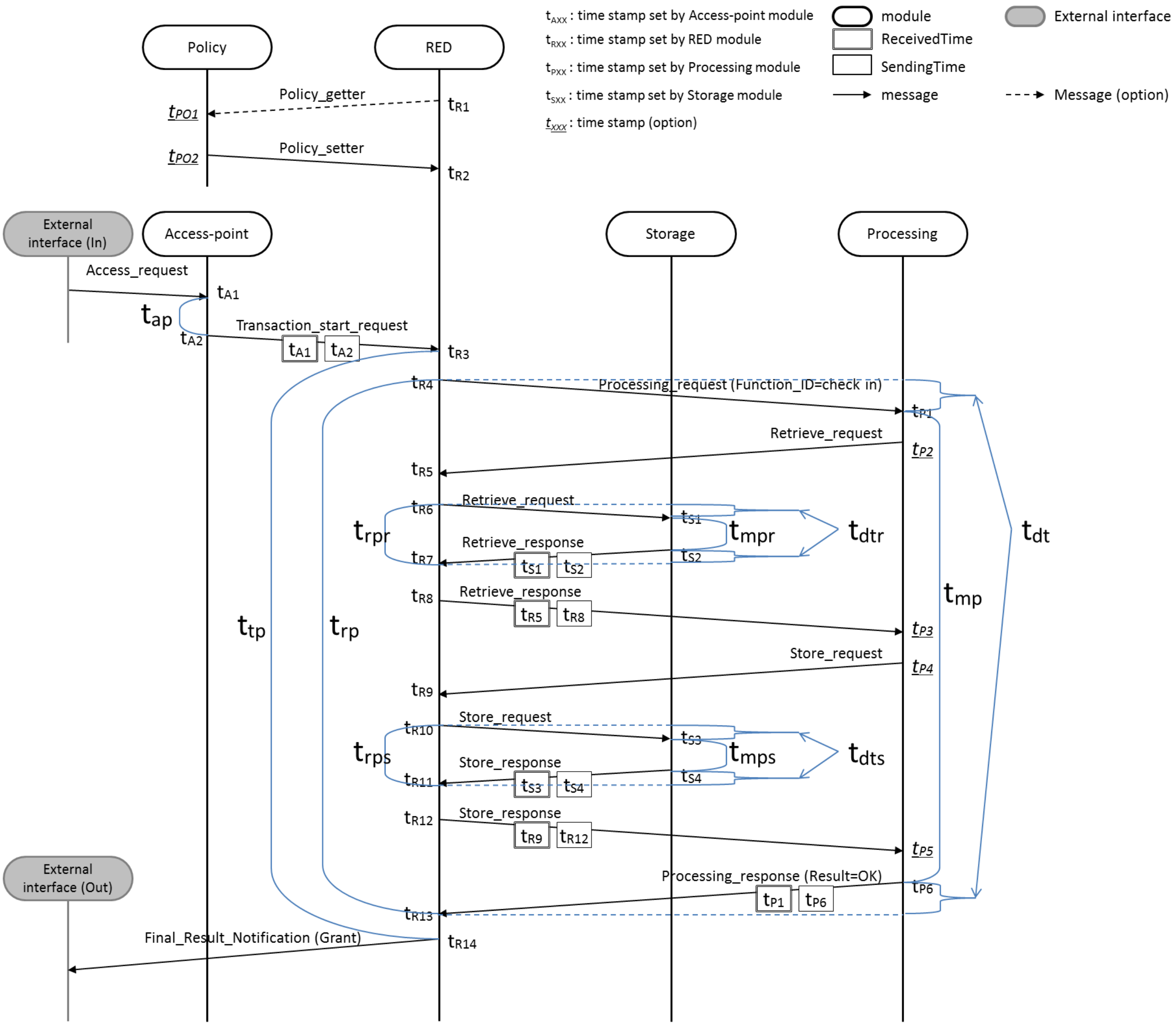

Figure C.1 — An example of time stamping and time measurement

(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}.