

**Standard** ECMA-412

2<sup>nd</sup> Edition / June 2017

**Framework for  
distributed real-time  
Access systems**

**Standard**



**COPYRIGHT PROTECTED DOCUMENT**

<b>Contents</b>	<b>Page</b>
1	<b>Scope</b> ..... 1
2	<b>Conformance</b> ..... 1
3	<b>Normative references</b> ..... 1
4	<b>Terms, definitions and acronyms</b> ..... 1
5	<b>Overview</b> ..... 3
6	<b>Transaction</b> ..... 3
7	<b>Time stamping function</b> ..... 4
8	<b>Module</b> ..... 4
8.1	<b>Policy module</b> ..... 5
8.2	<b>Access-point module</b> ..... 5
8.3	<b>RED module</b> ..... 5
8.4	<b>Processing module</b> ..... 6
8.5	<b>Storage module</b> ..... 6
9	<b>Messages of each interface</b> ..... 6
9.1	<b>Messages of Policy interface</b> ..... 7
9.2	<b>Message of Access interface</b> ..... 7
9.3	<b>Messages of Processing interface</b> ..... 7
9.4	<b>Messages of Storage interface</b> ..... 11
10	<b>Messages of external interfaces</b> ..... 12
10.1	<b>Access_request from external interface (In)</b> ..... 12
10.2	<b>Final_Result_Notification to external interface (Out)</b> ..... 13
10.3	<b>Time_stamp_Notification</b> ..... 13
11	<b>Access system performance management</b> ..... 14
11.1	<b>Transaction processing time</b> ..... 14
11.2	<b>Request performance time</b> ..... 15
11.3	<b>Module processing time</b> ..... 16
11.4	<b>Data transmission time</b> ..... 16
11.5	<b>Request performance time for retrieve</b> ..... 16
11.6	<b>Module processing time for retrieve</b> ..... 17
11.7	<b>Data transmission time for retrieve</b> ..... 17
11.8	<b>Request performance time for store</b> ..... 17
11.9	<b>Module processing time for store</b> ..... 18
11.10	<b>Data transmission time for store</b> ..... 18
11.11	<b>Access point processing time</b> ..... 19
	<b>Annex A (informative) Service access control system</b> ..... 21
	<b>Annex B (informative) Share information between different Access systems</b> ..... 23
	<b>Annex C (informative) Usage of time stamping</b> ..... 25



## 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 2<sup>nd</sup> 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 1<sup>st</sup> 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.

**"COPYRIGHT NOTICE**

© 2017 Ecma International

*This document may be copied, published and distributed to others, and certain derivative works of it may be prepared, copied, published, and distributed, in whole or in part, provided that the above copyright notice and this Copyright License and Disclaimer are included on all such copies and derivative works. The only derivative works that are permissible under this Copyright License and Disclaimer are:*

- (i) works which incorporate all or portion of this document for the purpose of providing commentary or explanation (such as an annotated version of the document),*
- (ii) works which incorporate all or portion of this document for the purpose of incorporating features that provide accessibility,*
- (iii) translations of this document into languages other than English and into different formats and*
- (iv) works by making use of this specification in standard conformant products by implementing (e.g. by copy and paste wholly or partly) the functionality therein.*

*However, the content of this document itself may not be modified in any way, including by removing the copyright notice or references to Ecma International, except as required to translate it into languages other than English or into a different format.*

*The official version of an Ecma International document is the English language version on the Ecma International website. In the event of discrepancies between a translated version and the official version, the official version shall govern.*

*The limited permissions granted above are perpetual and will not be revoked by Ecma International or its successors or assigns.*

*This document and the information contained herein is provided on an "AS IS" basis and ECMA INTERNATIONAL DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."*

# 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**

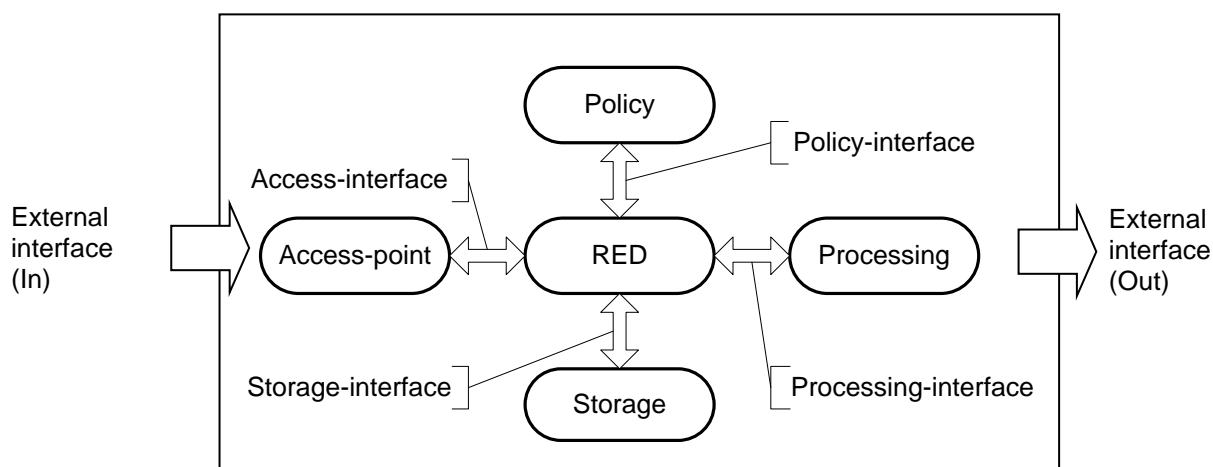
a request to initiate a transaction

## 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

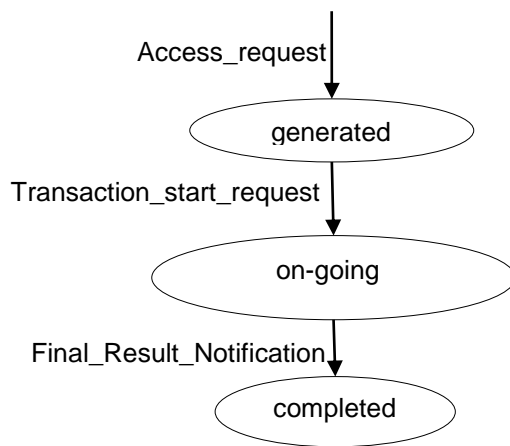
A transaction is a suite of functions and message exchanges to generate a final result and send it to a receiver. A transaction starts from the time an Access system receives an access request and completes after it sends the result.

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.

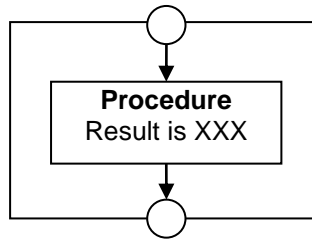
The rules should define:

- the destination and the timing of Time\_Stamp\_Notification.

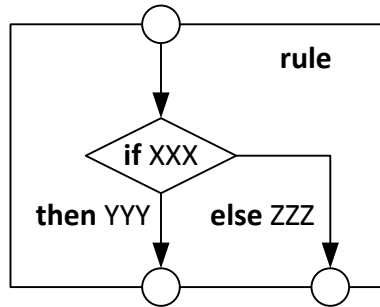
At least one rule is linked to Access ID.

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 through 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 Transaction\_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 through 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 Processing 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 to Retrieve\_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

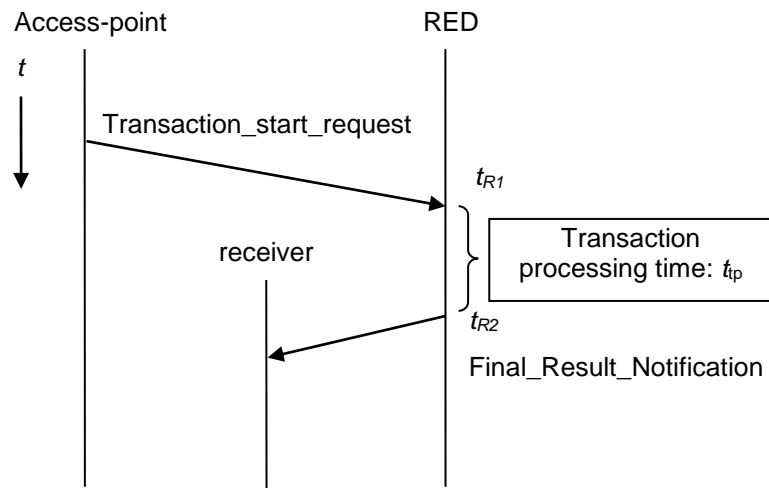
The choice of performance elements and metrics depends on application and service requirements. For example, the performance is specified "The maximum time (metric) of Transaction processing time (element) is less than x seconds (value)".

### 11.1 Transaction processing time

Transaction processing time ( $t_p$ ) shown in Figure 5 describes the performance of an Access system to process a transaction.  $t_p$  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)

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



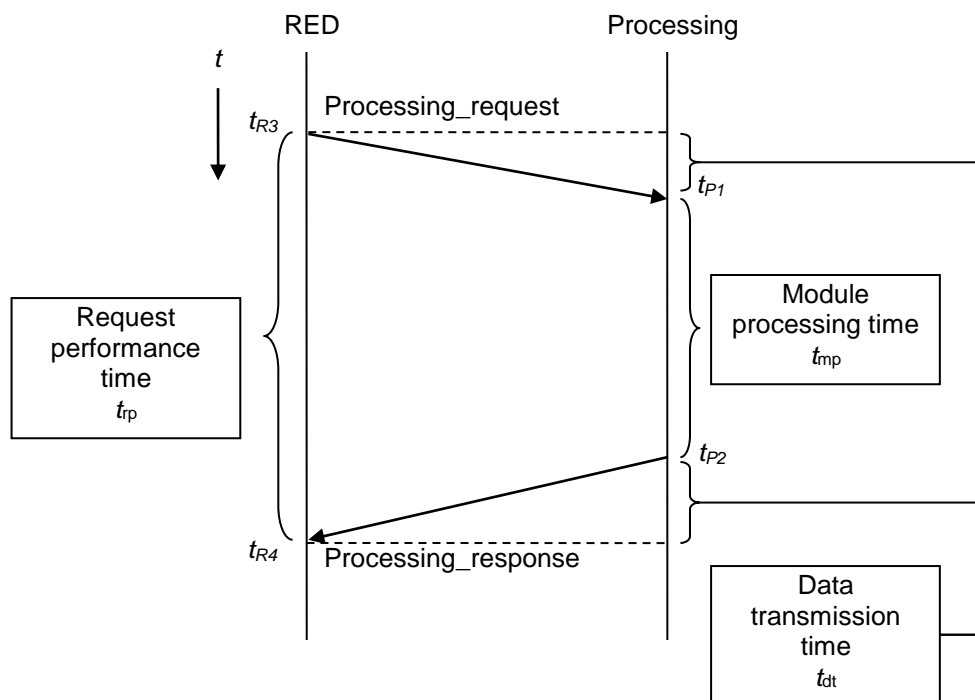
**Figure 5 — Transaction processing time**

## 11.2 Request performance time

Request performance time ( $t_p$ ) shown in Figure 6 describes the performance of the Processing module and the data transmission time.  $t_p$  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)

$$t_p = t_{R4} - t_{R3} \quad (11.2)$$



**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)

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

### 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  $t_{mp}$ .

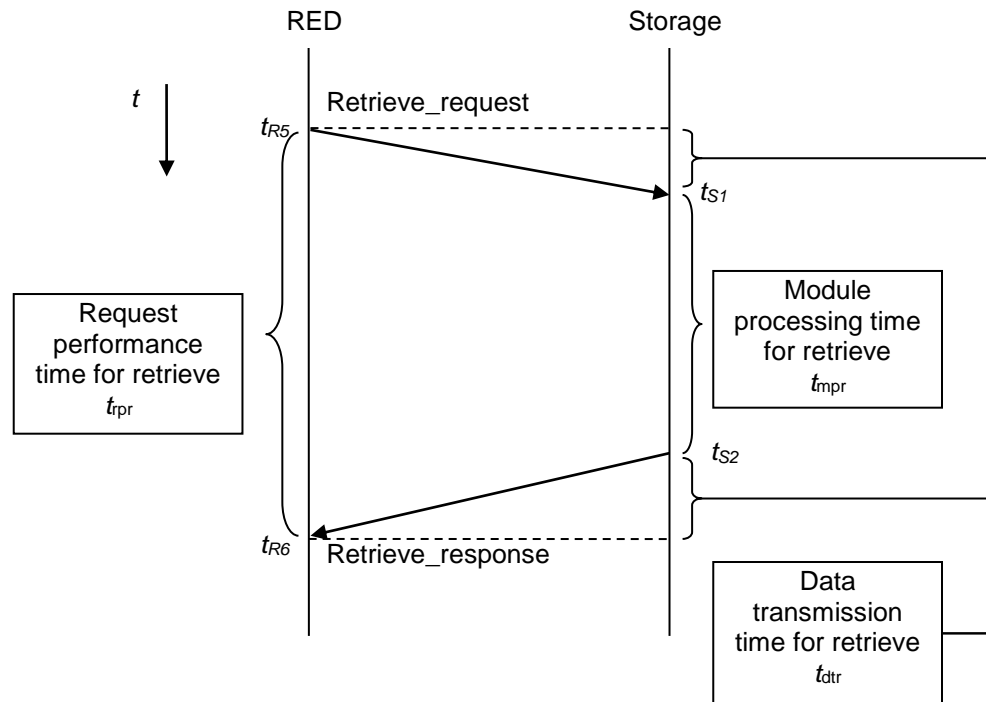
$$t_{dt} = t_{rp} - t_{mp} \quad (11.4)$$

### 11.5 Request performance time for retrieve

Request performance time for retrieve ( $t_{pr}$ ) shown in Figure 7 describes the performance of retrieving data from the Storage module.  $t_{pr}$  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)

$$t_{pr} = t_{R6} - t_{R5} \quad (11.5)$$



**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)

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

### 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  $t_{pr}$  and  $t_{mpr}$ .

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

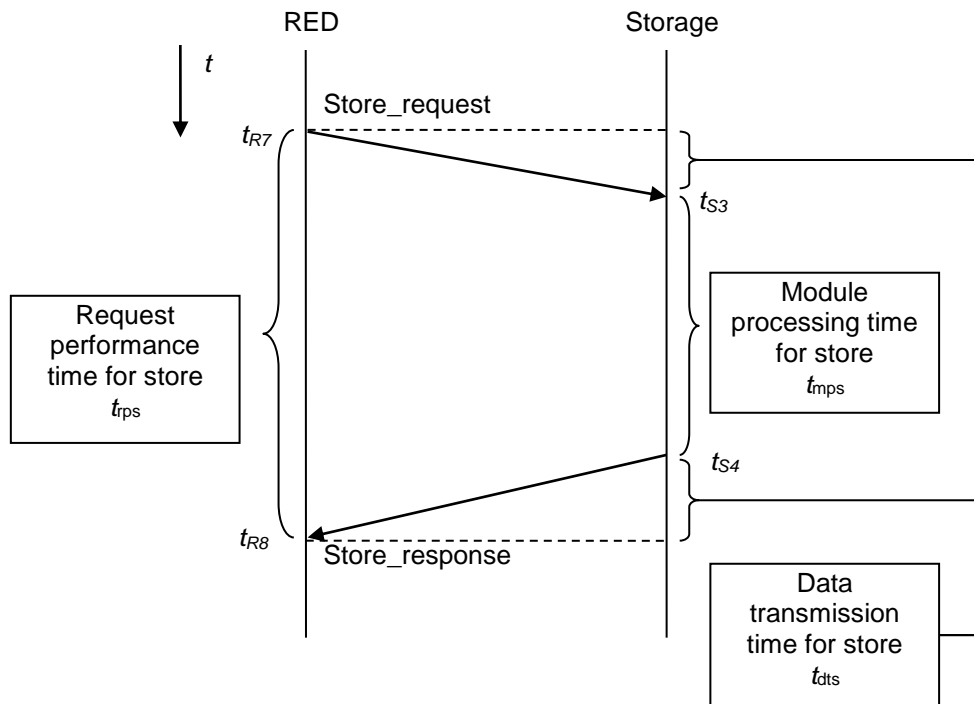
### 11.8 Request performance time for store

Request performance time for store ( $t_{rps}$ ) describes the performance of storing data from storage module.  $t_{rps}$  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 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)

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

### 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:

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



### 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)

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

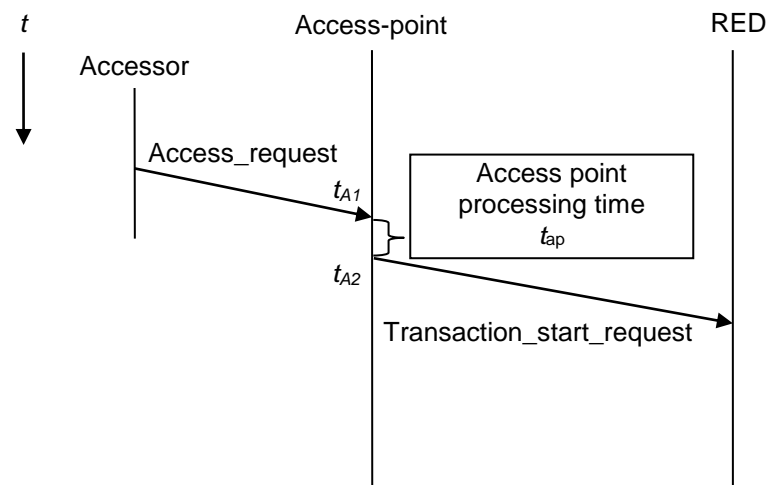


Figure 9 — Access point processing time



## 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 (shown in Figure A.1)

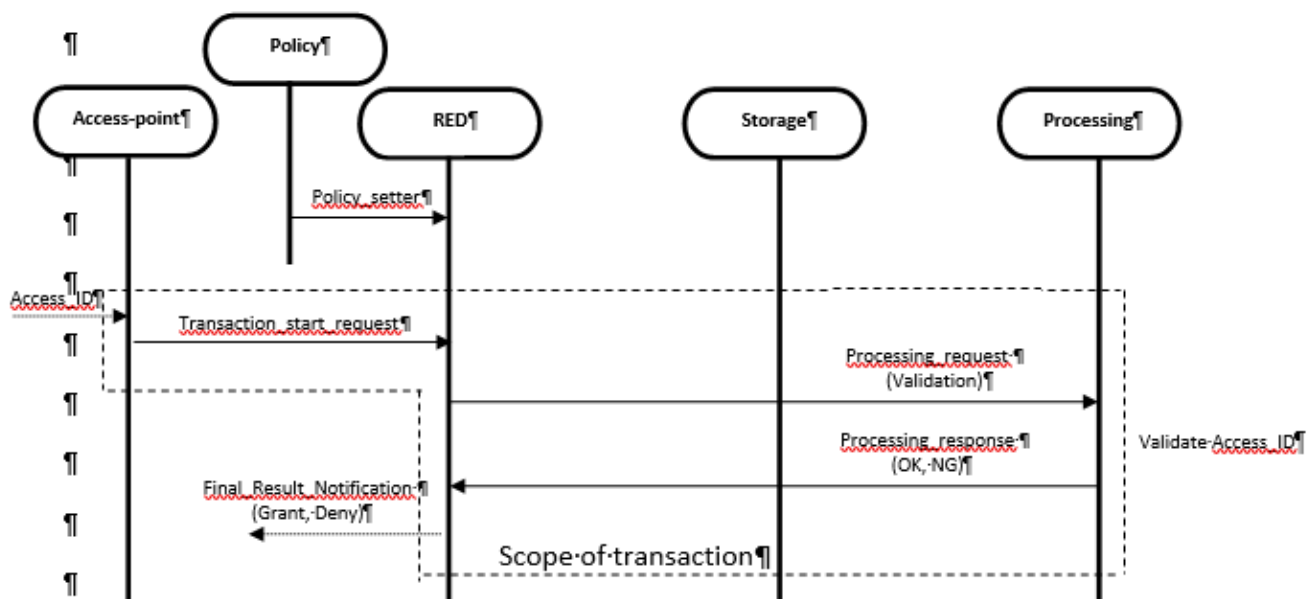
The Policy module sends Policy\_setter to the RED module to set the rules.

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**



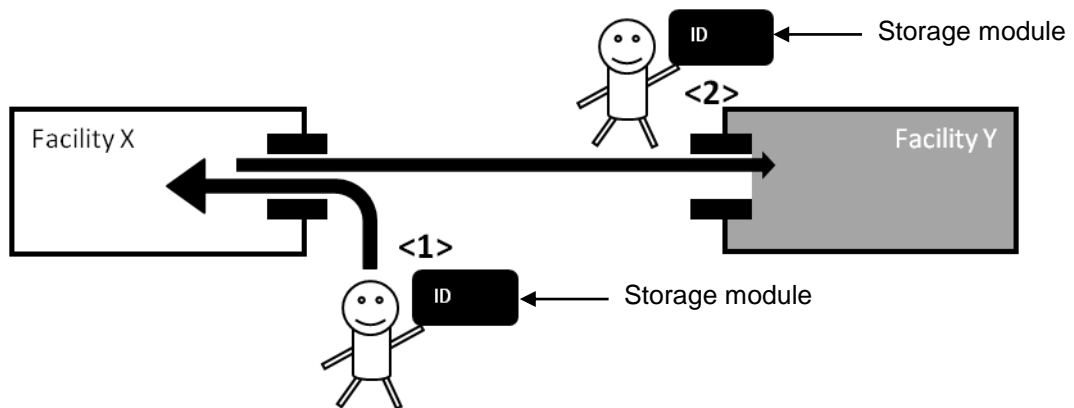
## Annex B (informative)

### 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 Y, 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>)

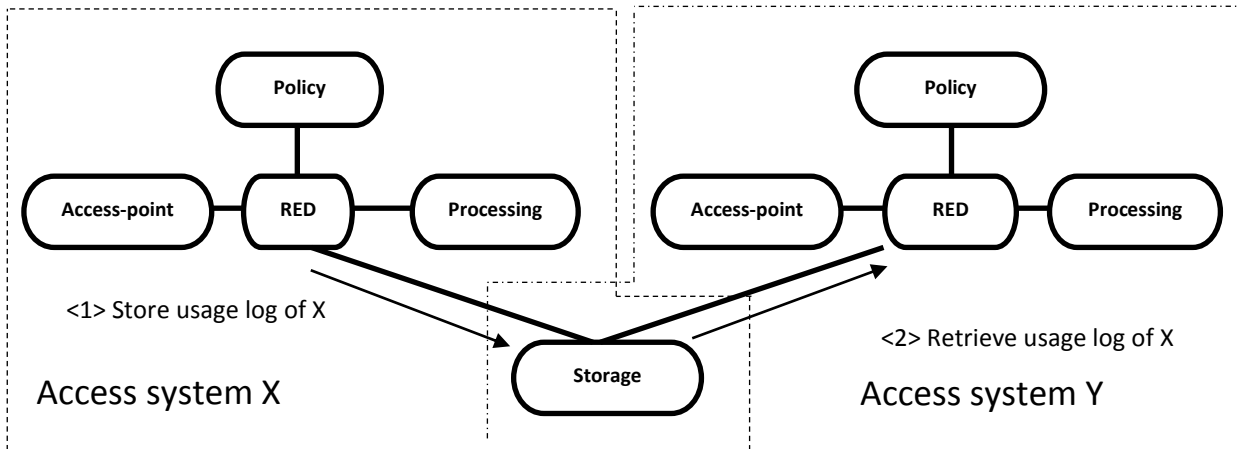


**Figure B.1 — An example of shared information application**

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

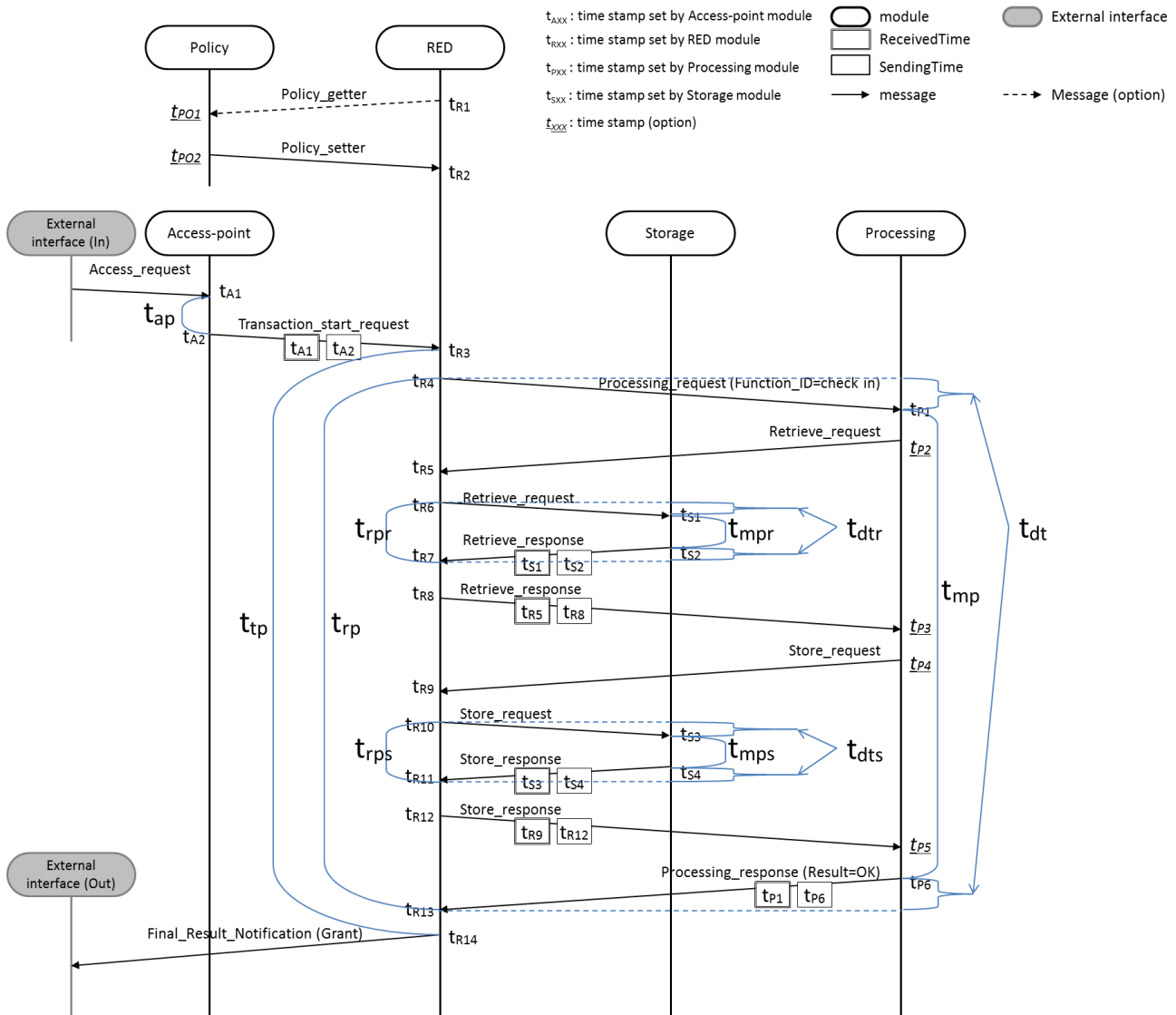


Figure C.1 — An example of time stamping and time measurement

- (1) RED module sends Policy\_getter at  $t_{R1}$ .
- (2) Policy module receives Policy\_getter at  $t_{P01}$ .
- (3) Policy module sends Policy\_setter at  $t_{P02}$ .
- (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  $t_{R14}$ .

