ecma INTERNATIONAL

Standard ECMA-412

1st Edition – June 2015

Access Systems

Standard

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

Contents

Introduction

Technology for real-time access control is widely used for many situations such as entrance gate 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 Ecma Standard has been adopted by the General Assembly of June 2015.

Access Systems

1 Scope

This standard specifies:

1) an ID triggered modular access system, the functions of the modules and the messages they exchange, and the sequence of messages, i.e. transitions of the transaction;

2) the system responsibility from receiving an access request until sending the result. i.e. a complete transaction;

3) the responsibilities of the modules, including time stamping and responding to the requests they received; and

4) the sequence and semantics of the messages and their elements.

2 Conformance

Conformant Access Systems progress transactions by evaluating 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
ID
Identifier

4.2
RED
Rule Evaluation and Dispatching

4.3
transaction
request for access

5 Model

Figure 1 illustrates the Access System structure.

The Access System has 5 modules "Access-point, Policy, Processing, RED and Storage" and 4 interfaces "Access-interface, Policy-interface, Processing-interface and Storage-interface".

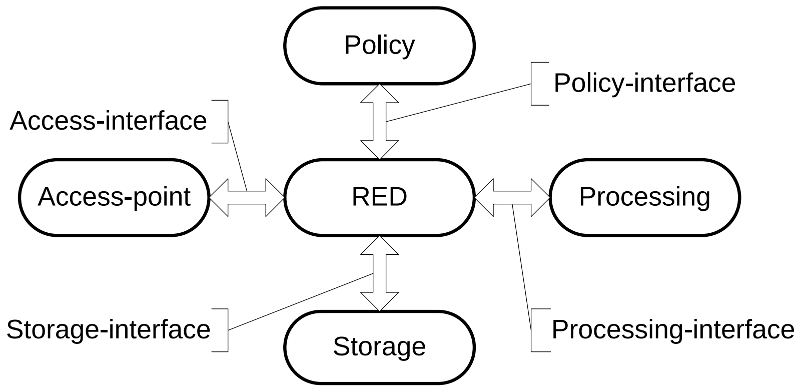


Figure 1 — Access System

The Access System progresses a transaction by exchanging messages between modules and decides the final result (grant or deny). A transaction starts when an Access-point module obtains Access_request and completes when the RED module sends Final_Result_Notification. Each module shall have a time stamping function. The message exchanging and the time stamping function are managed by the RED module according to rules which are set by the Policy module.

6 Transaction

Transaction ID identifies a transaction. Transaction ID shall consist of Access ID, Access-point ID and time at which the Access_request is obtained. Access ID is included in Access_request.

Figure 2 specifies the state machine of a transaction.

A transaction is generated at the time of Access_request acceptance by an Access-point module. After that the transaction changes to on-going state by sending a Transaction_start_request including Transaction ID from the Access-point module to the RED module.

At the on-going state, the RED module evaluates rules until final result is obtained. According to the result of the evaluation, the RED module sends a request message to Processing or Storage module and receives a response message.

When the RED module obtains the final result, it sends Final_Result_Notification and the transaction is completed.

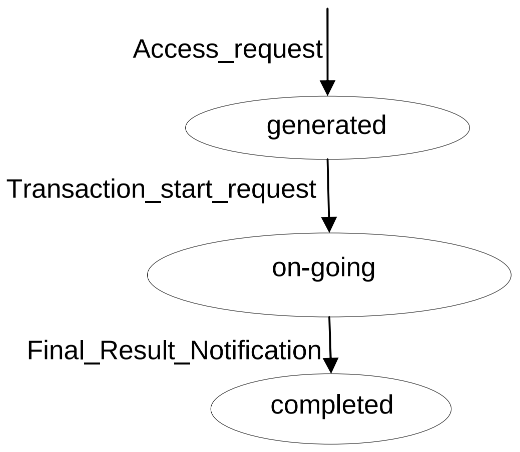


Figure 2 — Transaction State Machine

7 Time stamping function

The purpose of Time stamping function is to measure the duration of transaction and request processing.

The Access-point modules shall set the Access_ID_obtained_time in the Transaction_start_request message. For the other modules, time stamping shall be activated and deactivated through time stamping rules. Upon evaluating of the time stamping rules, the RED module shall set the TimeStampingFlag value in the requests to TRUE or FALSE according to the evaluation. Depending on the TimeStampingFlag value in the requests, modules shall either time stamp the ReceivedTime and SendingTime or exclude those elements in the corresponding response.

The RED module shall send the time stamping measurements by responding to the Time_stamp_Notification.

The RED module is able to measure following time.

1) transaction processing time

2) request processing time.

When the Time stamping function of each module is activated, the RED module shall measure the following time.

3) module processing time.

The RED module shall measure the transaction processing time by calculating the difference between the time that the RED module received Transaction_start_request and the time that Final_Result_Notification is sent.

The RED module shall measure the request processing time by recording the sending time of the request and the received time of the response, and calculating the difference between them.

Processing_response, Store_response and Retrieve_response have the information about the received time of the corresponding request and the sending time of the response itself as long as the Time stamping function is activated. By using them, the RED module is able to measure the module processing time. For example, the module processing time of the Processing module for one request from the RED module is measured by the difference between RecievedTime and SendingTime in the corresponding Processing_response.

Annex C illustrates the usage of time stamping.

8 Module

8.1 Common requirements

Modules shall have a time stamping function.

8.2 Policy module

The Policy module shall have the source of rules, and shall set the rules to the RED module. Each rule shall be identified by its Rule ID. The rules shall define the progress of transactions and the edition of this standard that the Access System modules conform with. And the rules shall identify the receiver(s) of the Final_Result_Notification and the receiver(s) of the Time_stamp_Notification.

8.3 Access-point module

When an Access-point module obtains an Access_request, It shall generate a Transaction_start_request and send it to the RED module.

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

8.4 RED module

The RED module shall accept and hold rules that are set by the Policy module.

Rules are composed of procedure rules and branch rules, 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. At least one rule is linked to Access ID.

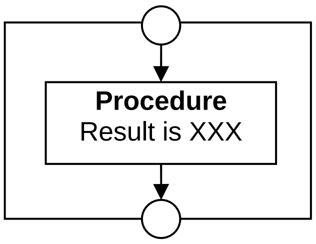


Figure 3 — procedure rule

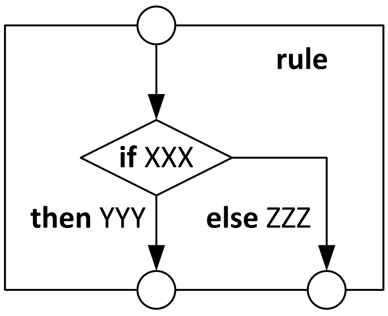


Figure 4 — branch rule

During a transaction, the RED module is driven by messages. When the RED module receives messages, It shall evaluate the rules. The RED module shall dispatch the request and response from one module to another according to the rules. When the result of the transaction is settled as grant or deny, the RED module shall send the Final_Result_Notification to the receiver(s) specified in the rules.

The RED module makes a Processing_request according to the rules and sends it to the Processing module. The RED module shall receive a Processing_response corresponding to the Processing_request.

When the RED module receives a Store_request from the Processing module, it shall transfer it to the Storage module. When the RED module receives a Retrieve_request from Processing module, the RED module shall transfer it to the Storage module. When the RED module receives Store_response from the Storage module, the RED module shall transfer it to the Processing module. When the RED module receives a Retrieve_response from Storage module, it shall transfer it to the Processing module.

To manage time stamping information, the RED module shall log 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) specified in the rules.

8.5 Processing module

The Processing module shall have at least one function. Each function shall be identified by its Function ID.

When the Processing module receives a Processing_request from the RED module, it shall execute the function identified by the Function ID in the request, make Processing_response including execution result and send it to the RED module.

The Processing module may request the RED module to store and retrieve data.

8.6 Storage module

When the Storage module receives Store_request, it shall store data, make Store_response and send it to the RED module. When the Storage module receives Retrieve_request, it shall retrieve data, make Retrieve_response including retrieved data and send it to the RED module. Data is specified by Data_type.

The Storage module may be used for sharing information between different transactions in the same Access System or different Access System as illustrated in Annex B; Annex A is an example use case that does not use Storage module.

9 Message definition and Interface

9.1 General

This clause specifies the messages exchanged via interfaces. Each message shall include the elements defined in clause 9 and may include other elements. In this document, the messages are specified by ASN.1 expression. Encoding rules are not specified.

Requests and responses include a Transaction ID. The type of Transaction ID is:

```
TransactionID_TYPE ::= SEQUENCE {
     Access_ID                   OCTET_STRING,
     Access-point_ID             OCTET_STRING,
     Access_ID_obtained_time GeneralizedTime
}
```

9.2 Policy interface

The Policy module shall use the Policy_setter to set the rules to the RED module and may use it at any time. The RED module may use the Policy_getter to request the rules at any time. The Policy_getter depends on implementations.

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

Policy_getter ::= SET {
      RULE_ID                   OCTET STRING
}
```

9.3 Access request

The Access-point module obtains the following message from an accessor.

```
Access_request ::= SET {
      Access_ID                 OCTET STRING
}
```

9.4 Access interface

The Access-point module shall use the following request to generate a new transaction.

```
Transaction_start_request ::= SET {
      Transaction_ID            TransactionID_TYPE
}
```

9.5 Processing interface

The RED module shall use the following request to execute the function according to the rule.

```
Processing_request ::= SEQUENCE {
      Transaction_ID            TransactionID_TYPE,
      Function_ID               OCTET STRING,
      TimeStampingFlag          BOOLEAN,
      SetOfParameter            SET {
                                Parameter    OCTET STRING
                                }
}
```

TimeStampingFlag shall indicate the information whether the Time stamping function is activated or deactivated.

The Processing module shall use the following corresponding response.

```
Processing_response ::= SEQUENCE {
      Transaction_ID            TransactionID_TYPE,
      Function_ID               OCTET STRING,
      ReceivedTime              GeneralizedTime,
      SendingTime               GeneralizedTime,
      Result                    OCTET STRING
}
```

Function_ID shall be the same data as Function_ID in the corresponding Processing_request.

Processing_response shall include ReceivedTime if TimeStampingFlag in the corresponding Processing_request is TRUE. ReceivedTime shall indicate the time at which Processing module received the corresponding Processing_request from RED module.

Processing_response shall include SendingTime if TimeStampingFlag in the corresponding Processing_request is TRUE. SendingTime shall indicate the time at which this response is sent.

Result shall include result of executing the function.

The Processing module may use the following requests to store and retrieve data.

```
Store_request ::= SEQUENCE{
        Transaction_ID              TransactionID_TYPE,
        Function_ID                 OCTET STRING,
        TimeStampingFlag            BOOLEAN,
        Data_type                   OCTET STRING,
        Data                        OCTET STRING
}
```

TimeStampingFlag shall be the same as TimeStampingFlag in the Processing_request that has the same Transaction_ID.

```
Retrieve_request ::= SEQUENCE{
        Transaction_ID              TransactionID_TYPE,
        Function_ID                 OCTET STRING,
        TimeStampingFlag            BOOLEAN,
        Data_type                   OCTET STRING
}
```

TimeStampingFlag shall be the same as TimeStampingFlag in the Processing_request that has the same Transaction_ID.

The RED module shall use the following responses.

```
Store_response ::= SEQUENCE{
        Transaction_ID              TransactionID_TYPE,
        Function_ID                 OCTET STRING,
        ReceivedTime                GeneralizedTime,
        SendingTime                 GeneralizedTime,
        Result                      OCTET STRING
}
```

Function_ID shall be the same as Function_ID in the corresponding Store_request.

Store_response shall include ReceivedTime if TimeStampingFlag in the corresponding Store_request is TRUE. RecievedTime shall indicate the time at which Storage module received the corresponding Store_request from RED module.

Store_response shall include SendingTime if TimeStampingFlag in the corresponding Store_request is TRUE. SendingTime shall indicate the time at which this response was sent.

Result shall indicate whether Data in the corresponding Store_request is stored or not.

```
Retrieve_response ::= SEQUENCE{
      Transaction_ID              TransactionID_TYPE,
      Function_ID                 OCTET STRING,
      ReceivedTime                GeneralizedTime,
      SendingTime                 GeneralizedTime,
      Data                        OCTET STRING
}
```

Function_ID shall be the same data as Function_ID in the corresponding Retrieve_request.

Retrieve_response shall include RecievedTime if TimeStampingFlag in the corresponding Retrieve_request is TRUE. RecievedTime shall indicate the time at which Storage module received the corresponding Retrieve_request from RED module.

Retrieve_response shall include SendingTime if TimeStampingFlag in the corresponding Retrieve_request is TRUE. SendingTime shall indicate the time at which this response is sent.

Data shall include the data which was retrieved upon the corresponding Retrieve_request.

9.6 Storage interface

The RED module may use the following requests to store and retrieve data.

```
Store_request ::= SEQUENCE{
      Transaction_ID              TransactionID_TYPE,
      Function_ID                 OCTET STRING,
      TimeStampingFlag            BOOLEAN,
      Data_type                   OCTET STRING,
      Data                        OCTET STRING
}
```

```
Retrieve_request ::= SEQUENCE{
      Transaction_ID              TransactionID_TYPE,
      Function_ID                 OCTET STRING,
      TimeStampingFlag            BOOLEAN,
      Data_type                   OCTET STRING
}
```

The Storage module shall use the following corresponding responses.

```
Store_response ::= SEQUENCE{
      Transaction_ID              TransactionID_TYPE,
      Function_ID                 OCTET STRING ,
      ReceivedTime                GeneralizedTime,
      SendingTime                 GeneralizedTime,
      Result                      OCTET STRING
}
```

```
Retrieve_response ::= SEQUENCE{
      Transaction_ID              TransactionID_TYPE,
      Function_ID                 OCTET STRING ,
      ReceivedTime                GeneralizedTime,
      SendingTime                 GeneralizedTime,
      Data                        OCTET STRING
}
```

Each element is the same as in 9.5 Processing interface.

9.7 Final result Notification

The RED module shall use the following Final_Result_Notification to notify the final result(grant or deny).

```
Final_Result_Notification :: = SEQUENCE{
    Transaction_ID          TransactionID_TYPE,
    ResultOfTransaction     ENUMERATED{ GRANT, DENY },
}
```

9.8 Time stamp Notification

The RED module shall use the following Time_stamp_Notification to output timestamp information. Timing of sending this message depends on the rules.

```
Time_stamp_Notification :: = SETOF{
     Transaction_ID             TransactionID_TYPE,
     TimeStampInformation       CHOICE{
                                     Transaction processing time     Generalized time,
                                     Request processing times        SET OF GeneralizedTime,
                                     Module processing times         SET OF GeneralizedTime
                                     }
}
```

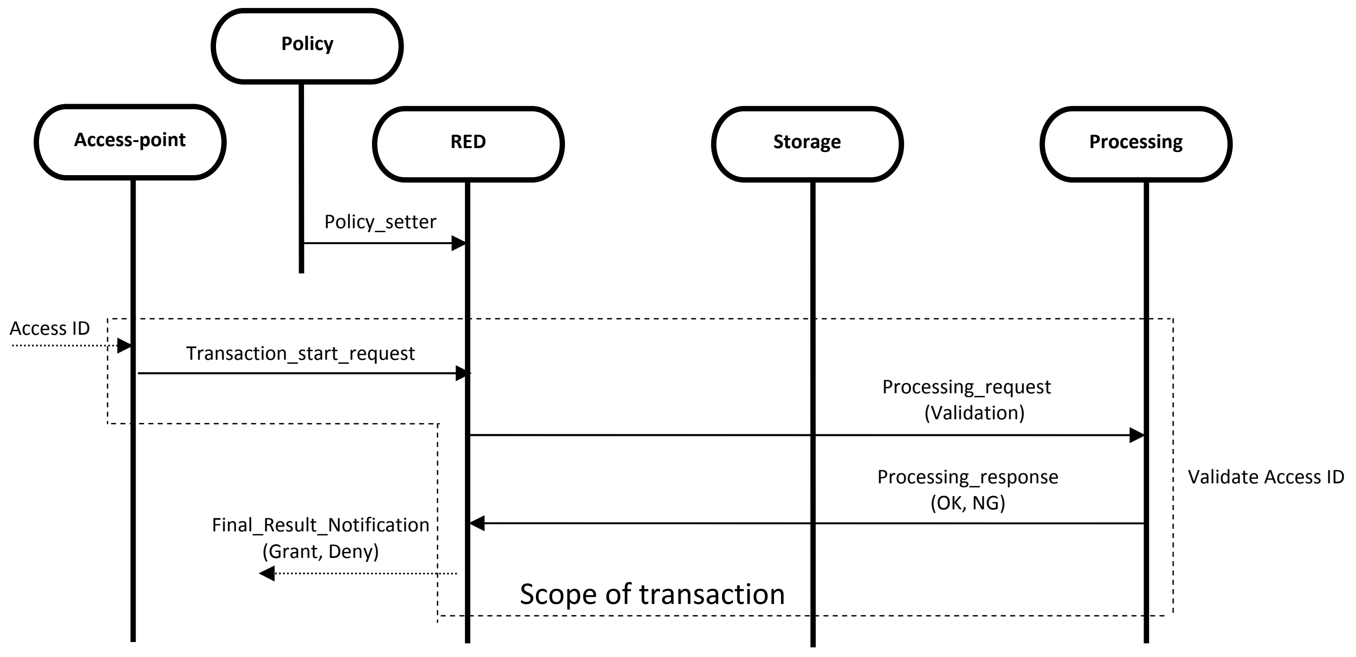
Annex A
(informative)

Service access control system

In this service access control system, users need some kind of authentication before using devices.

An Access System, a service access control system, provides such authentication process.

A typical example of message sequence is as follows:


Policy
Access-point
RED
Storage
Processing
Policy_setter
Access ID
Transaction_start_request
Processing_request (Validation)
Processing_response (OK, NG)
Validate Access ID
Final_Result_Notification (Grant, Deny)
Scope of transaction

The Policy module sends Policy_setter to the RED module to set the rules.

If the Access-point module obtains Access ID, then it sends a Transaction_start_request to the RED module.

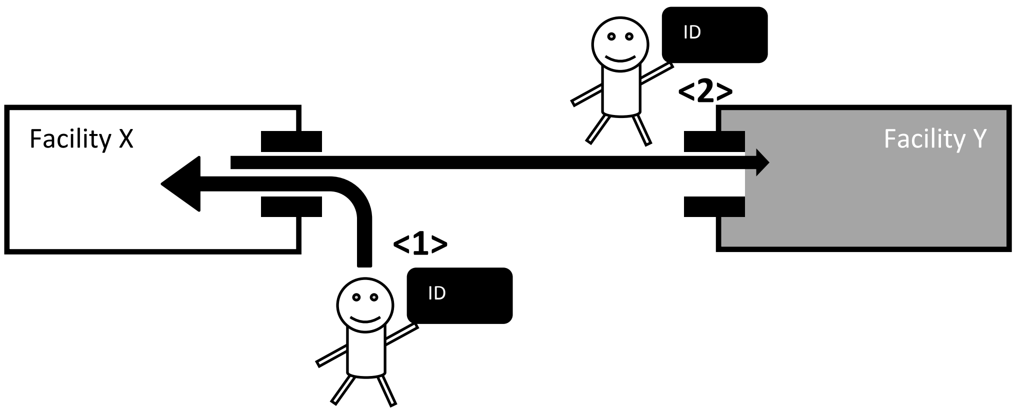
The RED module evaluates the rules and then it sends a Processing_request to the Processing module to execute a validation function.

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

The RED module sends a Final_Result_Notification including the final result (grant or deny) to the receiver.

Annex B
(informative)

Share information between different Access Systems

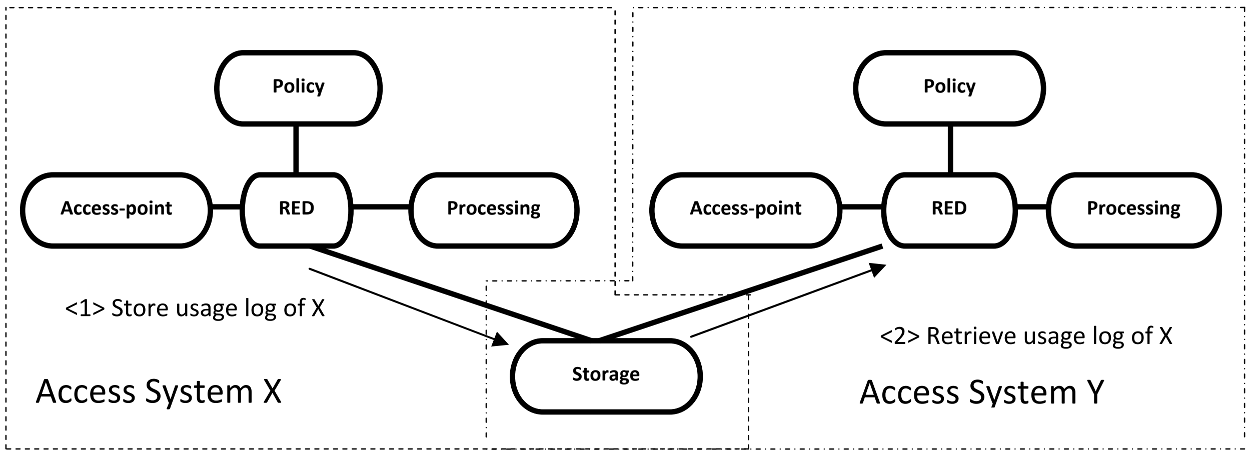


<1> User uses Facility X by his ID. Then usage log is stored in a Storage module which is shared between Access System X and Y.

<2> After using Facility X, user uses to Facility Y by the same ID. In order to provide combination service between Facility X and Y (e.g. discount, point program and so on), the usage log which is stored in shared Storage module is used by Facility Y.

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

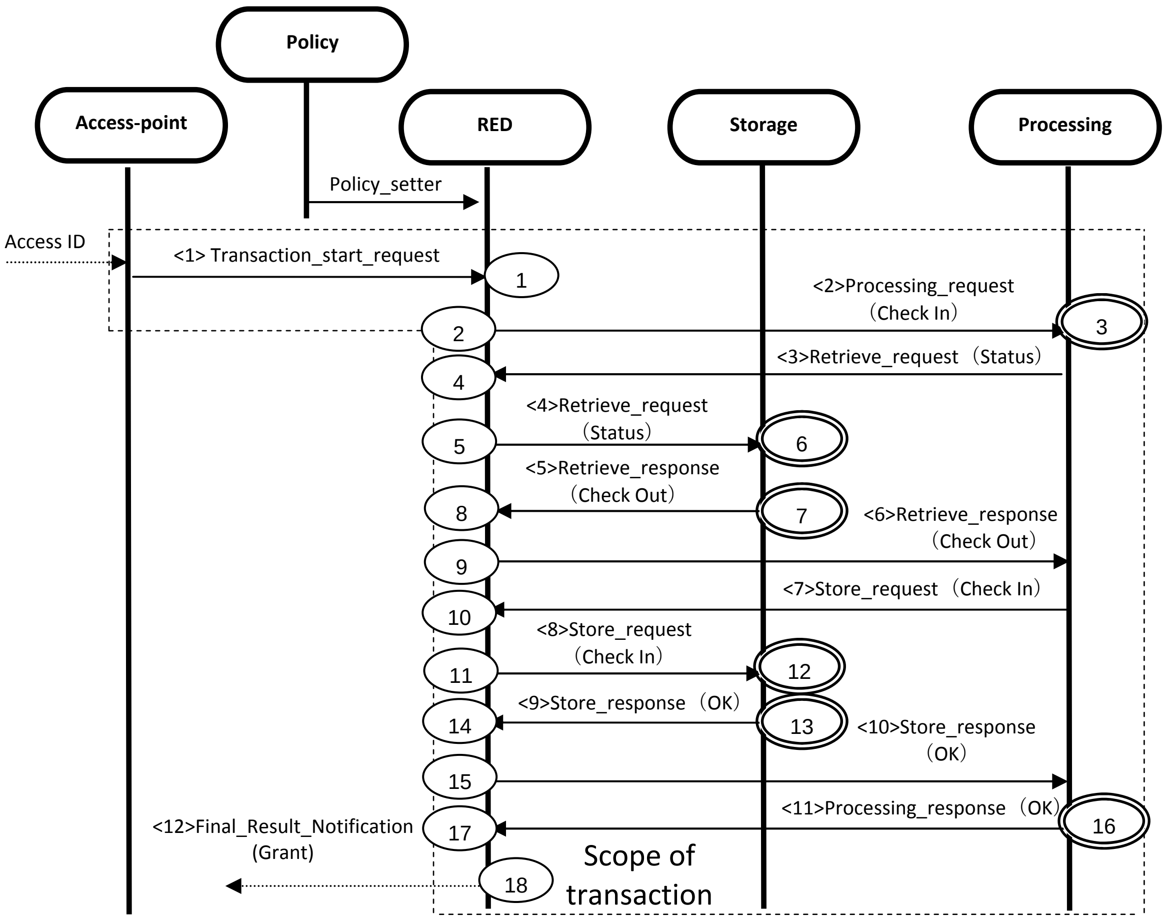
The example of two Access systems which have shared a Storage module is as follows:



Annex C
(informative)

Usage of Time_stamping

The RED module records the time-stamps at following points in the typical example of message sequence in Annex A.


Policy
Access-point
RED
Storage
Processing
Policy_setter
Access ID
<1> Transaction_start_request
1
<2>Processing_request (Check In)
2
3
<3>Retrieve_request (Status)
4
<4>Retrieve_request (Status)
5
6
<5>Retrieve_response (Check Out)
8
7
<6>Retrieve_response (Check Out)
9
<7>Store_request (Check In)
10
<8>Store_request (Check In)
11
12
<9>Store_response (OK)
14
13
<10>Store_response (OK)
15
<11>Processing_response (OK)
16
<12>Final_Result_Notification (Grant)
17
18
Scope of transaction

Recording point of time stamp

○ RED module records the time when a request is sent or a response is received

◎ Storage module or Processing module records the time when a request is received or a response is sent, and sends them to RED module by setting into a response message.

Each point shows following timing.

【1】RED module receives Transaction_start_request.

【2】RED module sends Processing_request.

【3】Processing module receives Processing_request.

【4】RED module receives Retrieve_request.

【5】RED module sends Retrieve_request.

【6】Storage module receives Retrieve_request.

【7】Storage module sends Retrieve_response.

【8】RED module receives Retrieve_response.

【9】RED module sends Retrieve_response.

【10】RED module receives Store_request.

【11】RED module sends Store_request.

【12】Storage module receives Store_request.

【13】Storage module sends Store_response.

【14】RED module receives Store_response.

【15】RED module sends Store_response.

【16】Processing module sends Processing_response.

【17】RED module receives Processing_response.

【18】RED module sends Final_Result_Notification.

RED module is able to measure the following duration times.

(1) **Transaction processing time（T1）=**【18】**-**【1】

(2) **Request processing time of Processing module（T2）=**【17】**-**【2】

(3) **Module processing time of Processing module（T3）=**【16】**-**【3】

(4) **Propagation delay on Processing_interface（T4）=T2 - T3**

Propagation delay for coming and going is able to measure by calculating T2－T3. So, rough propagation delay from RED module to Processing module or from Processing module to RED module is able to measure by calculating（T2－T3) / 2.

(5) **Request processing time of Storage module for retrieve（T5）=**【8】**-**【5】

(6) **Module processing time of Storage module for retrieve（T6）** =【7】-【6】

(7) **Propagation delay on Storage_interface for retrieve（T7）=T5 – T6**

Propagation delay for coming and going is able to measure by calculating T5－T6 . So, rough propagation delay from RED module to Processing module or from Processing module to RED module is able to measure by calculating（T5－T6) / 2.

(8) **Request processing time of Storage module for store（T8）** =【14】-【11】

(9) **Module processing time of Storage module for store（T9）** =【13】-【12】

(10) **Propagation delay on Storage_interface for store（T10）=T8 – T9**

Propagation delay for coming and going is able to measure by calculating T8－T9 . So, rough propagation delay from RED module to Processing module or from Processing module to RED module is able to measure by calculating（T8－T9) / 2.

(11) **Module processing time of Processing module except request processing time from Processing modules to RED module （T11）=T3 - (【9】-【4】+T4) - (【15】-【10】+T4)**.