ISO 10161-1: 1997 /DAM 1

Title: **Documentation - Interlibrary Loan Application Protocol Specification - Amendment 1: Support for Use of Object Identifier in "identifier" Parameter of the Extension Data Type**

Source: **Canada**

Date: **4 February 1998**

Status: **Approved CD**

Background

Implementation experience in North America has brought to light a weakness in the conceptual basis and specification of the Extension data type that is present in every ILL APDU.

The Extension type consists of three parameters, an "identifier", a flag to indicate whether support for the extension is "critical", and the "item" containing the extension itself. The *identifier* parameter is defined to be an integer, while the *item* parameter is specified as "ANY DEFINED BY" the *identifier*. This definition allows for any possible extension to be supported: the nature of the extension and the data type used to carry the information is implicitly identified by implementors' shared understanding of the meaning of individual values of the *identifier* integer. According to ISO 8824, however, the use of an integer as *identifier* requires that all permissible values be defined within the protocol standard itself, either in the base standard or in amendments. There is no mechanism for registering new values for *identifier* outside the standard for either experimental purposes or arising from stable implementors' agreements.

In order to allow experimentation, and to enable communities that require extensions to implement working systems without the serious delays that the international standardization process can impose, it is desirable to enable extensions to be defined using a registration mechanism outside the standard itself. Such mechanisms already exist for information objects identified using the object identifier mechanism, and this mechanism could readily be used for extensions to ILL APDUs. In order to enable such use, however, an amendment to the base ILL protocol standard is required.

This amendment is therefore intended to enable the use of externally registered objects in the *item* parameter of the Extension data type. There are two possible ways of implementing this. One way would be to make *identifier* a CHOICE between an integer and an object identifier. This approach, however, means that the ASN.1 specification of the base ILL APDUs would change, which would require that the version number of every APDU change. This seems undesirable for such a minor revision. It is therefore proposed that an alternative, but very simple, mechanism be used. That is, to define in the standard a value for *identifier* that indicates that the *item* is an external object. It is therefore proposed here that value 1 for *identifier* defines *item* to be a data type of EXTERNAL.

It should be noted that this approach offers slightly less flexibility than that of making *identifier* a CHOICE. If *identifier* were an object identifier, there would no requirement that *item* be an external object, and slightly simpler encodings could possibly be used. The advantage of avoiding a new protocol version, however, seems to outweigh this minor loss of flexibility.

Because a protocol amendment to define a value for *identifier* is being proposed here, error conditions will also have to be defined to allow a system to indicate that it does not support (or recognize) a value of *item*. Since the extensibility rules require a system to ignore an unknown value of a known data type, these error conditions will only be necessary in the case that an extension is flagged as *critical*. Three mechanisms for indicating the error are proposed here:

1. The error may be indicated as a Reason_Unfilled in an ILL-Answer APDU
2. The error may be indicated as a Provider-Error in a Status-Report-Or-Error APDU
3. The error may be indicated as a User-Error in a Status-Report-Or-Error APDU

The difference between mechanism 2 and 3 lies in the point at which the error is identified. If the external object is unknown to the receiving system, a provider error should be generated. If the external object is known, but not supported, a user error should be generated.

The current amendment does not preclude the possibility of proposing other amendments to define additional values for *identifier*.

Proposed Amendment to ISO 10161-1: 1997

Clause 9.1.2

Change the definition of the Extension data type (lines 589-593) to:

```
Extension ::= SEQUENCE {
        identifier      [0]     IMPLICIT INTEGER,
        -- value 1 = implementor extensions
        critical        [1]     IMPLICIT BOOLEAN DEFAULT FALSE,
        item            [2]     ANY DEFINED BY identifier
        -- if identifier = 1 then item ::= EXTERNAL
}
```

Add a value to General-Problem (lines 594-600):

```
General-Problem ::= ENUMERATED {
        ...
        other                                   (5),
        critical-extension-not-recognized       (6)
        }
```

Add a value to Intermediary-Problem (lines 693-695):

```
Intermediary-Problem ::= ENUMERATED {
        cannot-send-onward              (1),
        critical extension-not-supported (2)
        )
```

Add a value to Reason-Unfilled (lines 805-832):

```
Reason-Unfilled ::= ENUMERATED {
        ...
        preferred-delivery-not-possible (24),
```

critical extension not supported (25),

...

}

Add a value to Unable-To-Perform (lines 1026-1030):

Unable-To-Perform ::= ENUMERATED {

{ ...

other (3),

critical-extension-not-supported (4)

}

Annex B Clause B.6.2 Mapping ASN.1 Types to EDIFACT Segments

Add two segments in extension to permit the encoding for externally-defined extensions:

Segment Name: extension

Segment Code: EXT

Data elements: --none; this segment comprises other segments
--rather than actual data elements. It includes the
--following segments:
extension-identifier
extension-critical
extension-item-external-start
extension-item-external-finish
--this segment will also contain one or more
--other segments which encode the data elements
--i.e. of the type "item";
--the segments which may be included here are
--defined by the value of "extension-identifier"

Segment Name: extension-item-external-start

Segment Code: EXS

Data elements: syntax
--an object-identifier for the syntax of the object
--contained in the following segments up to EXF.
--this segment will be followed by other segments, not
--defined here, carrying the encoding of the externally-
--defined object identified by the syntax element.

Segment Name: extension-item-external-finish

Segment Code: EXF

Data elements: --none; this segment is used to indicate the end of
--the segments encoding an externally defined object.