ETSI TS 132 300 V9.2.0 (2010-07)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Configuration Management (CM);
Name convention for Managed Objects
(3GPP TS 32.300 version 9.2.0 Release 9)

Reference

RTS/TSGS-0532300v920

Keywords

GSM, LTE, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2010.
All rights reserved.

DECT™, **PLUGTESTS**™, **UMTS**™, **TIPHON**™, the TIPHON logo and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
LTE™ is a Trade Mark of ETSI currently being registered
for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

Configuration Management (CM), in general, provides the operator with the ability to assure correct and effective operation of the 3G network as it evolves. CM actions have the objective to control and monitor the actual configuration on the Network Elements (NEs) and Network Resources (NRs), and they may be initiated by the operator or by functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service (QoS). The CM actions are initiated either as single actions on single NEs of the 3G network, or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

Background

Traditionally, multiple name conventions have been used by different vendors' NEs, or even within the same vendor, to name network resources. The following problems have thus arisen:

- Different classes of NE have used different name conventions. Network Management applications, when interfacing with these NEs, have been required to understand multiple name conventions to manage the NEs.
- Network management applications (e.g. Fault Management application), when interfacing with other applications (e.g. Configuration Management application, trouble ticket system) have been required to understand multiple name conventions.
- When a customer purchased multiple classes of NEs from the same or different vendors, the customer was confronted with multiple name conventions.
- Without a name convention, it is difficult to integrate IRP conformant vendors' resource name space (see subclause 3.1.5 for definition of name space) into the customer's Enterprise name space.

Benefits

The benefits of using the subject name convention to name 3G network resources for network management purposes are as follows:

- A resource name is guaranteed to be unambiguous in that it refers to, at most, one network resource. Unambiguous naming of managed network resources is necessary for interoperability among managing applications and systems.
- The resource name syntax is specified such that management applications can be designed with assurance that its name-parsing algorithm needs not be modified in the future. We can derive this benefit only if the subject name convention is widely accepted.

The root and upper portions of the name hierarchy are based on name infrastructure of Domain Name System (DNS) (see IETF RFC 2247 [5]). The subject name convention can naturally fit in DNS and can integrate well with other hierarchical naming systems, such as ITU-T Recommendation X.500 [2].

1 Scope

A more detailed background and introduction of the IRP concept is given in 3GPP TS 32.150 [16].

To perform network management tasks, co-operating applications require identical interpretation of names assigned to network resources under management. Such names are required to be unambiguous as well. The present document recommends one name convention for network resources under management in the IRP context.

To facilitate integration of network management information obtained via multiple IRPs based on different IRP Solution Set technologies, identical network resource name semantics shall be conveyed in all IRPs. The present document specifies one such name convention.

The present document also specifies an IOC/MOC name recommendation (see annex E) in order to avoid potential problems with valid characters in some programming languages.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] Void.

[2] ITU-T Recommendation X.500 (1993): "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services".

[3] T. Howes, ISBN 1-57870-070-1: "Understanding and Deploying LDAP Directory Services".

[4] IETF RFC 1737 (1994): "Functional Requirements for Uniform Resource Names".

[5] IETF RFC 2247 (1998): "Using Domains in LDAP/X.500 Distinguished Names".

[6] IETF RFC 1035 (1987): "Domain names - implementation and specification".

[7] IETF RFC 2253 (1997): "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names".

[8] 3GPP TS 32.111-2: "Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

[9] 3GPP TS 32.622: " Telecommunication management; Configuration Management (CM); Generic network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[10] Void.

[11] 3GPP TS 32.101: "Telecommunication management; Principles and high level requirements".

[12] 3GPP TS 32.102: "Telecommunication management; Architecture".

[13] ISO/IEC 14977: "Information technology – Syntactic metalanguage – Extended BNF".

[14] ISO/IEC 646: "Information technology – ISO 7-bit coded character set for information interchange".

[15] ISO/IEC 10646: "Information technology – Universal multiple-octet Coded Character Set (UCS)".

[16] 3GPP TS 32.150: "Integration Reference Point (IRP) Concept and definitions".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TS 32.101 [11] , 3GPP TS 32.102 [12] and 3GPP TS 32.150 [16] apply. This subclause defines terms essential for understanding of name convention in the IRP context.

3.1.1 Void

3.1.2 Void

3.1.3 Managed Object and Network Resource

In the context of the present document, a Managed Object (MO) is a software object that encapsulates the manageable characteristics and behaviour of a particular network resource. Examples of network resource are switch, scanner for monitoring performance data, cell, site, transmission links, satellite, operator profile, etc. In the present document, MO sometimes is referred to as MO instance.

3.1.4 Name

In the context of the present document, a name is restricted to the identification of a MO, that is, a software object representing a real network resource.

3.1.5 Name space

A name space is a collection of names. This name convention uses a hierarchical containment structure, including its simplest form - the one-level, flat name space. This name convention does not support an arbitrarily connected name space, or graph structure, in which a named object can be both child and parent of another named object.
Figure 1 shows some examples of supported and unsupported name spaces (this figure is from T. Howes, ISBN 1-57870-070-1 [3] and it provides useful information on name space design).

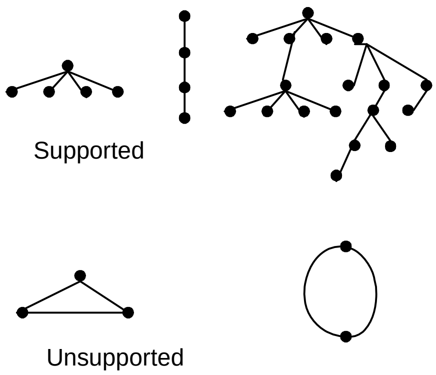

Figure 1: Examples of supported and unsupported name spaces

3.1.6 Global Root and Local Root

Names in name space are organised in hierarchy. An MO instance that contains another one is referred to as the superior (parent), whereas the contained MO instance is referred to as the subordinate (child).

In modern network management, it is expected that the Enterprise name space be partitioned for implementations in multiple managed system (see annex C for reasons of name space partitioning). The parent of all MO instances in a single managed system is called the Local Root. The ultimate parent of all MO instances of all managed systems is called the Global Root.

3.1.7 Distinguished Name and Relative Distinguished Name

A Distinguished Name (DN) is used to uniquely identify a MO within a name space. A DN is built from a series of "name components", referred to as Relative Distinguished Names (RDNs). ITU-T Recommendation X.500 [2] defines the concepts of DN and RDN in detail, using ASN.1, in the following way:

```
DistinguishedName ::= RDNSequence
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET SIZE (1..MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {type AttributeType, value AttributeValue}
```

The present document references this ASN.1 structure but it only uses single-valued (not multi-valued) RDN.

From a DN of a MO, one can derive the DN of its containing MO, if any. This containment relation is the only relation carried by the DN. No other relation can be carried or implied by the DN.

See annex B for a rule for MO designers to avoid ambiguity concerning the `AttributeType` of a DN string.

See annex C for discussion of DN prefix.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CM	Configuration Management
DC	Domain Component
DN	Distinguished Name
DNS	Domain Name Service
EBNF	Extended Backus-Naur Form
FM	Fault Management
IETF	Internet Engineering Task Force
IOC	Information Object Class
IRP	Integration Reference Point
IS	Information Service
LDN	Local Distinguished Name
MO	Managed Object
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NR	Network Resource
NRM	Network Resource Model
PM	Performance Management
QoS	Quality of Service
RDN	Relative Distinguished Name
SS	Solution Set
TMN	Telecommunications Management Network

4 System overview

4.1 System context

Situations under which MO (representing network resource) names are used are as follows:

a) MO names cross various Integration Reference Points (IRPs).

EXAMPLE 1: In the context of Alarm IRP 3GPP TS 32.111-2 [8], IRPAgent notifies IRPManager of the alarm condition of a network resource. The DN of the MO, representing alarmed network resource, encoded as specified in the present document, is carried in the `Managed Object Instance` parameter of the notification.

EXAMPLE 2: In the context of Generic Network Resources IRP: NRM, 3GPP TS 32.622 [9], IRPAgent notifies IRPManager of the creation of new object. The DN of the newly created object, encoded as specified in the present document, is carried in the notification.

EXAMPLE 3: In the context of Generic Network Resources IRP: NRM, 3GPP TS 32.622 [9], IRPManager requests IRPAgent to search for a particular object by specifying the start point of the search. The DN of the base object, upon which the search begins downward hierarchically, is carried in the request.

b) Co-operating management applications need to exchange information that includes MO (representing network resource) names.

EXAMPLE 4: A Fault Management (FM) application may request a trouble ticket system to open a new trouble ticket reporting the alarmed condition of a network resource by specifying, among other things, the MO name representing the alarmed network resource. The DN of the MO, encoded as specified in the present document, is included in the request.

EXAMPLE 5: A Performance Management (PM) system that produces reports on performance of network resources. The DNs of the MOs, representing the reported network resources, encoded as specified in the present document, are printed on the report.

5 Name Convention for Managed Objects

Network resources shall be named using the naming conventions in ITU-T Recommendation X.500 [2] with one restriction listed below. Central to the X.500 naming convention is the concept of the Distinguished Name (DN) (see subclause 3.1.7).

The restriction is that this IRP name convention does not support multi-value RDN. Only single-value RDN is supported.

6 Representations of Distinguished Name (DN)

A DN can be encoded and represented in many ways. The present document specifies one representation. Future work on IRP specifications may require the definition(s) of other representation(s)..

The DN is encoded using the string representation.

The DN string representation encoding scheme:

- shall be used for DNs exchanged through all IRP SS technologies,
- is in itself IRP SS technology neutral, and
- is subject to IRP SS technology specific handling, such as escaping, if required by such a technology.

7 String Representation of DN

7.A Overview

This clause specifies the string representation of DN. This work is based on IETF RFC 2253 [7]. A DN string representation, using the string-encoding scheme specified in the present document, is also a valid DN string according to IETF RFC 2253 [7].

The string-encoding scheme specified in the present document imposes further restrictions as compared to IETF RFC 2253 [7]. The most important restrictions are:

- Multi-valued RDN is not supported in the subject name convention.
- Character asterisk is used to denote wildcard in the subject name convention.

7.B Allowed character sets

Subject to further restrictions described in the following subclauses, the allowed characters for the string representation of DN are:

- Characters of ISO/IEC 646 [14] International Reference Version (IRV) coded character set, and
- Characters of standard coded character sets supporting and extending ISO/IEC 646 [14] IRV coded character set, e.g. ISO/IEC 10646 [15] coded character set.

NOTE 1: ISO/IEC 646 [14] IRV coded character set is the international equivalent to the ANSI X3.4 ASCII coded character set.

NOTE 2: The character set of ISO/IEC 646 [14] IRV corresponds to the subset of characters that range from U+0000 to U+007F in the character set of ISO/IEC 10646 [15].

NOTE 3: The ISO/IEC 646 [14] IRV characters specifically referenced in this specification are further identified using ISO/IEC 10646 [15] character short identifier notation form "U+XXXX".

7.1 Converting DN from ASN.1 to String

Subclause 7.1.1 defines the algorithm to convert an ASN.1 structured representation to one-string DN representation. Subclause 7.1.2 defines the algorithm to convert an ASN.1 structured representation to multi-string DN representation.

CORBA SS uses one-string DN representation. XML SS uses both one-string and multi-string DN representations.

7.1.1 Rule for one-string DN

7.1.1.1 Converting `RDNSequence`

If the `RDNSequence` is an empty sequence, the result is the empty or zero length string.

Otherwise, the output consists of the string encoding of each `RDN` in the `RDNSequence` (according to subclause 7.1.1.2), starting with the first element of the sequence and moving forward toward the last element.

The encoding of adjacent RDNs are separated by a comma character ('`,`', U+002C), to be consistent with IETF RFC 2253 [7].

White spaces adjacent to the comma character shall be ignored.

7.1.1.2 Converting RelativeDistinguishedName

When converting from an RDN to a string, the output consists of the string encoding of the singleton AttributeTypeAndValue (according to subclause 7.1, i.e. 'Multi-valued RDN is not supported in the subject name convention.').

Although ITU-T Recommendation X.500 DN supports multi-valued RDN, this specification supports single-valued RDN only.

7.1.1.3 Converting AttributeTypeAndValue

The AttributeTypeAndValue is encoded as the string representation of the AttributeType, followed by an equals sign character ('=', U+003D), followed by the string representation of the AttributeValue.

Although ITU-T Recommendation X.500 ASN.1 AttributeValue and AttributeType support wide range of character representation, this specification supports a restrictive set of characters according to subclause 7.2.

String representation of AttributeValue allows character escape mechanism such as the use of a reverse solidus character ('\', U+005C) followed by two hexadecimal digits to replace a character in a string. String representation of AttributeType does not allow character escape mechanism.

EXAMPLE: "CN=Before\0DAfter,O=Test,C=GB". In this example, the reverse solidus character and the two hexadecimal digits form a single byte in the code of the escaped character. The reverse solidus character followed by "0D" indicates a carriage return character. See annex B for a rule for MO designers to avoid ambiguity concerning the AttributeType of a DN string.

7.1.2 Rule for multi-string DN

7.1.2.1 Converting RDNSequence

If the RDNSequence is an empty sequence, there is no equivalent multi-string DN output, i.e. there is no multi-string DN representation for empty RDNSequence sequence.

Otherwise, the output consists of the string encoding of all RDNs in the RDNSequence (according to subclause 7.1.2.2).

One RDN encoding is within an NRM class associated XML element. A multiple RDNSequence would be converted into multiple RDN encodings, using multiple NRM class associated XML elements. They are arranged in a hierarchy in the XML document. The outer NRM class associated XML element represents the first RDN in the RDNSequence.

7.1.2.2 Converting RelativeDistinguishedName

When converting from an RDN to a string, the output consists of the encoding of the singleton AttributeTypeAndValue (according to subclause 7.1, i.e. 'Multi-valued RDN is not supported in the subject name convention.').

Although ITU-T Recommendation X.500 DN supports multi-valued RDN, this specification supports single-valued RDN only.

7.1.2.3 Converting AttributeTypeAndValue

Within an NRM class XML element, there is an element such as <element name="ManagedElement"> and there is another element <attribute name="id" type="string" use="required"/> which is inherited by using <extension base="xn:NrmClass">.

The AttributeType and AttributeValue are mapped to the former and latter XML elements. The AttributeType carries the IOC name while the AttributeValue carries the value of the IOC naming-attribute.

Although ITU-T Recommendation X.500 ASN.1 `AttributeValue` and `AttributeType` support wide range of character representation, this specification supports a restrictive set of characters according to subclause 7.2.

String representation of `AttributeValue` allows character escape mechanism (see 7.1.1.3).

7.2 Character syntax

This subclause specifies the character syntax for `AttributeType` and `AttributeValue`.

They are:

1. Any character except:
   - comma character (',', U+002C),
   - equals sign character ('=', U+003D),
   - carriage return character (U+000D),
   - line feed character (U+000A),
   - plus sign character ('+', U+002B),
   - less-than sign character ('<', U+003C),
   - greater-than sign character ('>', U+003E),
   - number sign character ('#', U+0023),
   - semicolon character (';', U+003B),
   - reverse solidus character ('\', U+005C),
   - quotation mark character ('"', U+0022).
2. The full stop character ('.', U+002E). This character shall be used in the `AttributeValue` whose `AttributeType` is "`DC`". An example is "`DC=marketing.CompanyXYZ.com`". This full stop character shall not be used in `AttributeType`.
3. The asterisk character ('*', U+002A) is reserved to denote wildcard. Wildcard character(s) can appear in `AttributeType` and `AttributeValue`. The wildcard character can be used to represent one or more characters.

7.3 EBNF of DN String Representation

The formal definitions provided within this subclause consolidate several rules and concepts (null distinguished name, DN prefix, local DN, domain component type, class names starting with upper case characters, attribute names starting with lower case characters, classes with or without an "Id" naming attribute, attribute type and attribute value allowed characters, wildcard character). The definition is more detailed to clarify these naming rules, and will not introduce compliancy issues for implementations compliant with Rel-5 version of this specification.

The following is the EBNF for DN in string representation (Extended Backus-Naur Form; see ISO/IEC 14977 [13] for more information):

```
DistinguishedName            =   NullDN       (* Distinguished Names shall not exceed   *)
                               | RegularDN ;  (* 400 octets as specified in section 7.4 *)

NullDN                       = ;  (* empty string; null DN is specified in subclause 7.1.1 *)

RegularDN                    =   DNPrefixPlusRDNSeparator  (* DN prefix and local DN *)
                               , LocalDN ;                 (* are defined in annex C *)

DNPrefixPlusRDNSeparator     =   ( NullDNPrefix , NullRDNSeparator )
                               | ( DNPrefixWithDomainComponent , RDNSeparator )
                               | ( DNPrefixWithoutDomainComponent , RDNSeparator ) ;
```

```
NullDNPrefix                   = ;  (* empty string *)

NullRDNSeparator               = ;  (* empty string *)

DNPrefixWithDomainComponent    =   DomainComponentRDN
                                 , { RDNSeparator , DomainComponentRDN }
                                 , { RDNSeparator , RegularRDN } ;

DNPrefixWithoutDomainComponent =   RegularRDN
                                 , { RDNSeparator , RegularRDN } ;

LocalDN                        =   LocalRDN
                                 , { RDNSeparator , LocalRDN } ;

RDNSeparator                   =   [ RDNSeparatorWhiteSpace ]     (* use of optional white space  *)
                                 , CommaChar                      (* is recommended to be avoided *)
                                 , [ RDNSeparatorWhiteSpace ] ;

RDNSeparatorWhiteSpace         =   [ CarriageReturnChar ]
                                 , { SpaceChar } ;

DomainComponentRDN             = DCAttributeTypeAndValue ;

RegularRDN                     = RegularAttributeTypeAndValue ;

LocalRDN                       = LocalDNAttributeTypeAndValue ;

DCAttributeTypeAndValue        =   DCAttributeType
                                 , AttributeTypeAndValueSeparator
                                 , ( DCAttributeValue | WildcardDCAttributeValue ) ;

RegularAttributeTypeAndValue   =   ( RegularAttributeType | WildcardRegularAttributeType )
                                 , AttributeTypeAndValueSeparator
                                 , ( RegularAttributeValue | WildcardRegularAttributeValue ) ;

LocalDNAttributeTypeAndValue   =   ( LocalDNAttributeType | WildcardLocalDNAttributeType )
                                 , AttributeTypeAndValueSeparator
                                 , ( RegularAttributeValue | WildcardRegularAttributeValue ) ;

AttributeTypeAndValueSeparator = EqualsSignChar ;

DCAttributeType                = "DC" ;  (* ISO/IEC 646 IRV U+0044/0043 Latin capital letters D&C *)

DCAttributeValue               =   DCLabel                           (* this is specified *)
                                 , { DCLabelSeparator , DCLabel } ;  (* in IETF RFC 1035  *)

WildcardDCAttributeValue       =   (    ( DCLabel | WildcardDCLabel )
                                      , { DCLabelSeparator , ( DCLabel | WildcardDCLabel ) } )
                                 - DCAttributeValue ;

DCLabelSeparator               = FullStopChar ;  (* this is specified in IETF RFC 1035 *)

DCLabel                        =   LetterChar                             (* this is specified *)
                                 , [   { LetterDigitHypenMinusChar }      (* in IETF RFC 1035  *)
                                     , LetterDigitChar               ] ;

WildcardDCLabel                =   (    ( LetterChar | WildcardChar )
                                      , [   { LetterDigitHypenMinusChar | WildcardChar }
                                          , ( LetterDigitChar | WildcardChar )           ] )
                                 - DCLabel ;

RegularAttributeType           =   LetterChar                        (* this is specified *)
                                 , { LetterDigitHypenMinusChar } ;   (* in IETF RFC 2253  *)

WildcardRegularAttributeType   =   (    ( LetterChar | WildcardChar )
                                      , { LetterDigitHypenMinusChar | WildcardChar } )
                                 - RegularAttributeType ;

LocalDNAttributeType           =   NameOfClassWithIdAttribute        (* definition selected shall *)
                                 | NamesOfClassAndNamingAttribute ;  (* be in accordance with the *)
                                                                     (* rules defined in annex B  *)

WildcardLocalDNAttributeType   =   WildcardNameOfClassWithIdAttr
                                 | WildcardNamesOfClassAndNamAttr ;

NameOfClassWithIdAttribute     = ClassName ;  (* see rules defined in annex B *)
```

```
WildcardNameOfClassWithIdAttr  = WildcardClassName ;

NamesOfClassAndNamingAttribute =   ClassName                      (* see rules defined in annex B *)
                                 , ClassNamingAttributeSeparator
                                 , NamingAttributeName ;

WildcardNamesOfClassAndNamAttr =   (   ( ClassName | WildcardClassName )
                                     , ClassNamingAttributeSeparator
                                     , ( NamingAttributeName | WildcardNamingAttributeName ) )
                                 - NamesOfClassAndNamingAttribute ;

ClassNamingAttributeSeparator  = FullStopChar ;  (* see rules defined in annex B *)

ClassName                      =   CapitalLetterChar               (* see recommendation on       *)
                                 , { LocalDNAttributeTypeChar } ;  (* characters for class names  *)
                                                                   (* in annex E
WildcardClassName              =   (   ( CapitalLetterChar | WildcardChar )
                                     , { LocalDNAttributeTypeChar | WildcardChar } )
                                 - ClassName ;

NamingAttributeName            =   SmallLetterChar
                                 , { LocalDNAttributeTypeChar } ;

WildcardNamingAttributeName    =   (   ( SmallLetterChar | WildcardChar )
                                     , { LocalDNAttributeTypeChar | WildcardChar } )
                                 - NamingAttributeName ;

RegularAttributeValue          =   ( AttributeValueChar - SpaceChar )           (* this is       *)
                                 , [   { AttributeValueChar }                   (* specified in  *)
                                     , ( AttributeValueChar - SpaceChar ) ] ;   (* IETF RFC 2253 *)

WildcardRegularAttributeValue  =   (   ( ( AttributeValueChar - SpaceChar ) | WildcardChar )
                                     , [   { AttributeValueChar | WildcardChar }
                                         , ( ( AttributeValueChar - SpaceChar ) | WildcardChar ) ] )
                                 - RegularAttributeValue ;

LocalDNAttributeTypeChar       = DNChar - FullStopChar ;

AttributeValueChar             = DNChar | EscapedCharSequence ;

WildcardChar                   = AsteriskChar ;  (* this is specified in subclause 7.2 *)

DNChar                         = DNCharUnrestricted - ReservedChar ;

DNCharUnrestricted             =   ? Character of ISO/IEC 646 IRV ?
                                 | ? Character of standard coded character set
                                     supporting and extending ISO/IEC 646 IRV  ? ;

EscapedCharSequence            =   ReverseSolidusChar           (* this is specified  *)
                                 , 2 * HexadecimalDigitChar ;   (* in subclause 7.1.3 *)

ReservedChar                   =   Rfc2253ReservedChar | CarriageReturnChar | LineFeedChar
                                 | AsteriskChar ;

Rfc2253ReservedChar            =   CommaChar | EqualsSignChar | PlusSignChar | LessThanSignChar
                                 | GreaterThanSignChar | NumberSignChar | SemiColonChar
                                 | ReverseSolidusChar | QuotationMarkChar ;

LetterChar                     = CapitalLetterChar | SmallLetterChar ;

LetterDigitChar                = LetterChar | DigitChar ;

LetterDigitHypenMinusChar      = LetterDigitChar | HypenMinusChar ;

HexadecimalDigitChar           = DigitChar | CapitalLetterAtoFChar | SmallLetterAtoFChar ;

LineFeedChar                   = ? ISO/IEC 646 IRV U+000A character line feed ? ;

CarriageReturnChar             = ? ISO/IEC 646 IRV U+000D character carriage return ? ;

SpaceChar                      = ' ' ;  (* ISO/IEC 646 IRV U+0020 character space *)

QuotationMarkChar              = '"' ;  (* ISO/IEC 646 IRV U+0022 character quotation mark *)

NumberSignChar                 = '#' ;  (* ISO/IEC 646 IRV U+0023 character number sign *)

AsteriskChar                   = '*' ;  (* ISO/IEC 646 IRV U+002A character asterisk *)
```

```
PlusSignChar                = '+' ;  (* ISO/IEC 646 IRV U+002B character plus sign *)

CommaChar                   = ',' ;  (* ISO/IEC 646 IRV U+002C character comma *)

HypenMinusChar              = '-' ;  (* ISO/IEC 646 IRV U+002D character hyphen-minus *)

FullStopChar                = '.' ;  (* ISO/IEC 646 IRV U+002E character full stop *)

DigitChar                   =   '0' | '1' | '2' | '3' | '4'    (* ISO/IEC 646 IRV U+0030-0039 *)
                              | '5' | '6' | '7' | '8' | '9' ;  (* digits 0 to 9               *)

SemiColonChar               = ';' ;  (* ISO/IEC 646 IRV U+003B character semicolon *)

LessThanSignChar            = '<' ;  (* ISO/IEC 646 IRV U+003C character less-than sign *)

EqualsSignChar              = '=' ;  (* ISO/IEC 646 IRV U+003D character equals sign *)

GreaterThanSignChar         = '>' ;  (* ISO/IEC 646 IRV U+003E character greater-than sign *)

CapitalLetterAtoFChar       =   'A' | 'B' | 'C'    (* ISO/IEC 646 IRV U+0041-0046  *)
                              | 'D' | 'E' | 'F' ;  (* Latin capital letters A to F *)

CapitalLetterChar           =   CapitalLetterAtoFChar | 'G' | 'H'    (* ISO/IEC 646 IRV *)
                              | 'I' | 'J' | 'K' | 'L' | 'M' | 'N'    (* U+0041-005A     *)
                              | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T'    (* Latin capital   *)
                              | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;  (* letters A to Z  *)

ReverseSolidusChar          = '\' ;  (* ISO/IEC 646 IRV U+005C character reverse solidus *)

SmallLetterAtoFChar         =   'a' | 'b' | 'c'    (* ISO/IEC 646 IRV U+0061-0066 *)
                              | 'd' | 'e' | 'f' ;  (* Latin small letters a to f  *)

SmallLetterChar             =   SmallLetterAtoFChar | 'g' | 'h'      (* ISO/IEC 646 IRV *)
                              | 'i' | 'j' | 'k' | 'l' | 'm' | 'n'    (* U+0061-007A     *)
                              | 'o' | 'p' | 'q' | 'r' | 's' | 't'    (* Latin small     *)
                              | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;  (* letters a to z  *)
```

7.4 Maximum size of DN string

The maximum length of a DN string, including RDN separators and including white spaces, shall not exceed 400 bytes (8-bit).

8 Examples of DN in string representation

This subclause gives a few examples of DN written in the string representation specified in the present document.

EXAMPLE 1: "`DC=com,DC=CompanyXYZ,DC=marketing,IRPAgent=ATMPVCBilling, Log=19990101131000,AccountingRecord=100098`". In this example, the name space aligns with DNS. The `AttributeType` of the top three RDN are "`DC`". Concatenation of the corresponding `AttributeValues` produces the DNS registered name, i.e. "`marketing.CompanyXYZ.com`". The top RDN is the Global Root because DNS defines "`DC=com`" as the root of its name space. That top RDN is the Local Root as well.

EXAMPLE 2: "`DC=marketing.CompanyXYZ.com,IRPAgent=ATMPVCBilling, Log=19990101131000,AccountingRecord=100098`". In this example, the name space aligns with DNS as well. Instead of using three RDNs to represent the DNS registered name, this example chooses to use one RDN. The top RDN is the Global Root (and Local Root as well).

EXAMPLE 3: "`IRPNetwork=ABCNetwork,Subnet=TN2,BSS=B5C0100`". In this example, the name space designer chooses not to name its objects under the DNS nor X.500 scheme. The name space designer chooses to use "`IRPNetwork=ABCNetwork`" as the Local Root of its name space (by looking at the DN string, it is not possible to say if the Local Root is the Global Root). DNs in this name space will start with that string as their Local Root. One string ("`IRPNetwork`") for `AttributeType` (of the `AttributeTypeAndValue` of the RDN) starts with "IRP". This indicates that this string is mapped from the MO class names specified in NRM of [9] or other domain specific NRMs (see the Introduction clause). Other strings do not start with "IRP", indicating that those strings are not mapped from MO class names specified in NRM of [9] or other domain specific NRMs. They are probably mapped from MO classes that are specific for a particular product and thus specified in a product-specific NRM.

EXAMPLE 4: The following example illustrates the use of the comma character as separator for RDNs. It also illustrates the use of space and full stop characters as part of the legal character syntax for RDNs: "`CN=John T. Mills, O=Cyber System Consulting`"

9 Usage Scenario

9.1 DN prefix usage

This subclause presents recommended steps designer uses to partition the Enterprise name space while building an Alarm IRP compliant NE (the Alarm IRP Agent).

1. The NE designer specifies the NRM (e.g. 3GPP TS 32.622 [9]) for the NE. Suppose the NRM is a two level hierarchy with 3 classes like:

```
Node
     |----- Port
     |----- CrossConnect
```

2. The NE designer, based on the NRM and other design choices, decides that there are 7 instances within the NE that can report alarms, such as

```
Port=1, Port=2, Port=3, Port=4, Port=5, CrossConnect=1, Node=1.
```

3. The NE designer decides on the DN prefix (see annex C) and configures its system accordingly. Since NE designer will not know the customer's name space in advance, he would normally configure the DN prefix to reflect his test environment. The DN prefix can be configured to "`Network=test`". The Global Root is "`Network=test`". The Local Root is "`Node=1`". It should be noted that the NE should not hard code the DN prefix but should treat DN prefix as a system configuration parameter, settable.

EXAMPLE 1: At system start-up time.

4. When constructing the alarm record (in coding phase), NE designer shall concatenate the name of the alarmed instance with the DN prefix to form the DN of his test environment. The resultant DN (e.g. "`Network=test,Node=1,Port=3`") will be placed in the `Managed Object Instance` (`MOI`) field of the alarm record.

5. The NE is sold to a customer. The customer administrator knows his Enterprise name space, the topology of his network and where the NE will be deployed. Based on the information, he configures the DN prefix of the NE.

EXAMPLE 2: The customer administrator can configure it to:

"`DC=marketing.CompanyXYZ.com,Net=DS3BackBone,Station=TMR`"

The Global Root in this case is "`DC=marketing.CompanyXYZ.com`".

6. At run time, whenever NE is reporting an alarm on `Port=3` via the IRP, the following string will be in the `MOI` field of the alarm record:

"`DC=marketing.CompanyXYZ.com,Net=DS3BackBone,Station=TMR,Node=1,Port=3`"

Annex A (normative): Mapping of RDN `AttributeType` to Strings

NOTE: This annex is normative for users of string representation.

`AttributeType` of `RDN` are mapped into strings for use in the DN string representation. This annex specifies the mapping.

The `AttributeType` shall include all MO classes defined in the Network Resource Model (NRM) of 3GPP TS 32.622 [9] and other domain specific NRMs as listed in the Introduction clause.

There is one `AttributeType` that is not defined in NRM of 3GPP TS 32.622 [9] or other domain specific NRMs as listed in the Introduction clause. This special `AttributeType` is used to denote the domain component of the DNS. The following partial DN string representations are examples to illustrate the valid use of "`DC`" strings for the three DNS domain components of "`marketing.CompanyXYZ.com`":

- "`DC=com.CompanyXYZ.marketing,…`"
- "`DC=com,DC=CompanyXYZ,DC=marketing,…`"
- "`DC=com,DC=CompanyXYZ.marketing,…`"
- "`DC=com.CompanyXYZ,DC=marketing,…`"

Table A.1: Example of RDN `AttributeType` Strings

String	AttributeType
DC	Domain component of DNS
SubNetwork	MO class name SubNetwork defined in NRM of 3GPP TS 32.622 [9].
etc.	See note.
NOTE: For each MO class name found in 3GPP set of specifications, its corresponding `AttributeType` String shall be identical to the class name with the leading character capitalised.	

Annex B (normative): Rule for MO Designers regarding `AttributeType` interpretation

NOTE: This annex is normative for users of one string representation.

This annex discusses the two possible interpretations for the `AttributeType` of the DN string and establishes a rule for MO designers to avoid ambiguity concerning their usage.
It also gives a rule for designing MO classes such that one DN string, regardless of the IRP SS technology used, will result in a unique reference to the corresponding network resource.

First interpretation

ITU-T Recommendation X.500 [2] uses the `AttributeType` (defined for use as the first component of the `AttributeTypeAndValue` of a RDN, see subclause 3.1.6) to identify one attribute of the subject MO for naming purpose. This `AttributeType` is called the *naming attribute* to distinguish itself from other attributes that may be present in the MO.

Suppose the following is the MO class definition in pseudo notation and this MO class is inherited from root.

```
Class Bsc {
  Attribute id;
  Attribute ..}
```

Suppose further that the naming attribute is `id`.

If this (first) interpretation is used for constructing the DN string, then the DN will be "`…,id=123`". MO class name cannot be derived from the DN string. The value of the `AttributeValue` contains the value of the naming attribute.

Second interpretation

Generally, this interpretation shall be preferred to the first interpretation.

The `AttributeType` (defined for use as the first component of the `AttributeTypeAndValue` of a RDN) is used to identify the MO class.Using this interpretation for constructing the DN string the DN will be "`…,Bsc=123`".
The name of the naming attribute cannot be derived from the DN string.
The value of the `AttributeValue` contains the value of the naming attribute.

Rule

Given the two interpretations, a DN reader cannot know how to interpret the `AttributeType,` i.e. if the `AttributeType` identifies class or naming attribute. To avoid ambiguity, the following rules shall apply:

- If the IS name of the IOC naming attribute is "id", ignoring case of 'id', then the DN shall use "`…,Yyy=123,…`" where "`Yyy`" is the IS name of the IOC, preserving case of 'Yyy'.

EXAMPLE 1: If "`Bsc`" is the IS name of the IOC and if the IS name of its naming attribute is "id' then the DN shall be "`…,Bsc=123,…`".

- If the IS name of the IOC naming attribute is not the concatenation of the IS name of the IOC and "Id", ignoring case for both, and if the IS name of the IOC naming attribute is not "id", ignoring case of 'id', then the DN shall use "`…,Yyy.zzz=123,…`" where "`Yyy`" is the IS name of the IOC and "`zzz`" is the IS name of the IOC naming attribute, preserving case for both.

EXAMPLE 2: If "`Bsc`" is the IS name of the IOC and if the IS name of its naming attribute is "`serialNumber`", then the DN shall be "`…,Bsc.serialNumber=123,…`".

- If the IS name of the IOC naming attribute is the concatenation of the IS name of the IOC and "Id", ignoring case for both, then the DN shall use "`…,Xxx=123,…`" where "`Xxx`" is the IS name of the IOC, preserving case.

EXAMPLE 3: If "`Bsc`" is the IS name of the IOC and if the IS name of its naming attribute is "`bscId`", then the DN shall be "`…,Bsc=123,…`".

Annex C (informative): DN Prefix and Local Distinguished Name (LDN)

A Distinguished Name (DN) is used to uniquely identify a MO within a name space. A DN is built from a series of "name components", referred to as Relative Distinguished Names (RDNs).

DNs within a name space are arranged in hierarchy similar to concepts of naming files in UNIX file system. A file name, in the context of a local subdirectory, contains the path (series of subdirectory names) of the file starting from the local subdirectory. The same file, in the global context, contains the path of the file starting from the root directory. Similar concept applies to naming MOs. From a particular (local) context, the name of a MO is the Local Distinguished Name (LDN). From a global context, the name of the same MO is the DN. LDN is a proper subset of DN. In the context of a particular local context, a DN prefix is defined such that all LDNs in that particular context, if attached behind the DN prefix of that context, will yield the DNs of the MOs.

The concepts of DN Prefix and LDN support the partitioning of large name space into smaller ones for efficient name space implementation. DN design, the subject of the present document, does not depend on these concepts. There exist other concepts that support partitioning of large name space as well. Although these concepts are independent from DN design, their use is wide spread and this annex illustrates their use in partitioning large name space.

In modern network management, it is expected that the Enterprise name space be partitioned for implementations in multiple hosts. The following are reasons for the partitioning.

- The Enterprise name space can be large (e.g. containing millions of objects). Partition of a large name space facilitates name space management.

EXAMPLE 1: It may be easier to manage two name spaces of 1 million objects each than to manage one name space with two million objects.

- Separate IRPAgents manage sub-set of the Enterprise name space relevant to their own local environment.

EXAMPLE 2: One NE manages a name space (subset of the Enterprise name space) containing names of its MOs representing its own network resources. Another NE manages another sub-set, etc.

- For reasons such as security, replication, back-up policy and performance, sub-sets of the Enterprise name space are managed by separate systems.

EXAMPLE 3: Operation and Marketing departments may want to manage their name spaces using their respective management policies. Partitioning of Enterprise name space according to departmental jurisdiction may facilitate deployment of independent management policies.

Suppose the Enterprise name space is organized hierarchically and is partitioned into 4 sub-sets as shown in figure C.1.

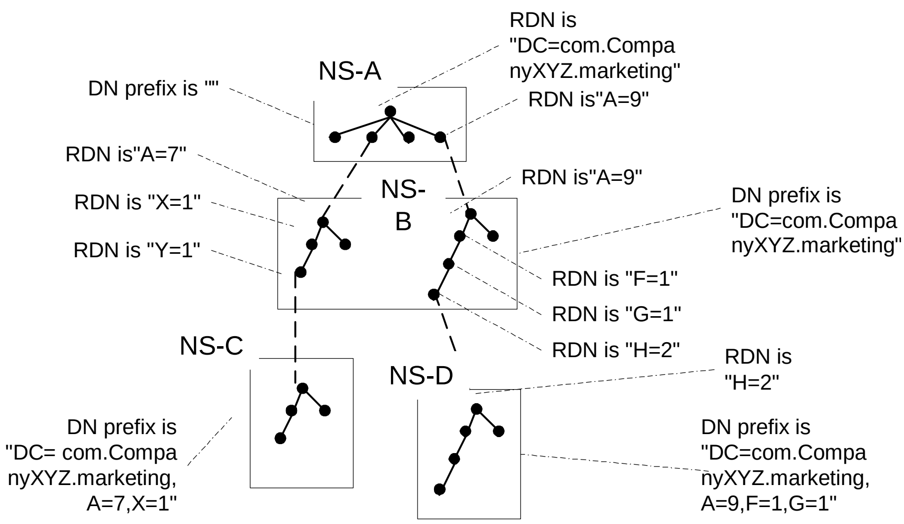

Figure C.1: Name space partitions

NS (name space)-A contains 5 objects. DN prefix is NULL. The Global Root and Local Root of NS-A is "DC=com.CompanyXYZ.marketing" (see the Note below). DN of top object is "DC=com.CompanyXYZ.marketing". RDNs of the other four objects are, from bottom left to bottom right, "A=1", "A=7", "A=3" and "A=9". DNs of the same four objects are "DC=com.CompanyXYZ.marketing,A=1", "DC=com.CompanyXYZ.marketing,A=7", "DC=com.CompanyXYZ.marketing,A=3" and "DC=com.CompanyXYZ.marketing,A=9". The second and fourth objects are reference objects to MOs in NS-B.

NS-B contains two branches. They have the same DN prefix that is "DC=com.CompanyXYZ.marketing". The Global Root is "DC=com.CompanyXYZ.marketing".

The Local Root and RDN of top object of the right branch is "A=9". Its DN is "DC=com.CompanyXYZ.marketing,A=9". RDNs of other objects are shown in figure C.1.
DN of the bottom object is "DC=com.CompanyXYZ.marketing,A=9,F=1,G=1,H=2". This object refers to object of another name space called NS-D.

The Local Root and RDN of the top object of the left branch is "A=7". Its DN is "DC=com.CompanyXYZ.marketing,A=7". RDNs of other objects are shown in figure C.1.
DN of the bottom object is "DC=com.CompanyXYZ.marketing,A=7,X=1,Y=1". This object refers to object of another name space called NS-C.

NS-C contains a branch of 4 objects. Its DN prefix is "DC=com.CompanyXYZ.marketing,A=7,X=1". The Local Root an RDN of the top object is "Y=1".

NS-D contains a branch of 5 objects. Its DN prefix is "DC=com.CompanyXYZ.marketing,A=9,F=1,G=1". The Local Root and RDN of the top object is "H=2".

In figure C.1, the bottom object of NS-B right branch has the following names:

- DN is "DC=com.CompanyXYZ.marketing,A=9,F=1,G=1,H=2".
- LDN is "A=9,F=1,G=1,H=2".
- RDN is "H=2".

With this example, we can see that DN of an object is a series of RDNs spanning the global name space. LDN of an object is a series of RDNs spanning the local name space where the subject MO resides.

The concatenation of the LDN with DN prefix of that (partitioned) name space shall produce the DN of the global name space.

NOTE: Use of "DC" in "DC=com.CompanyXYZ.marketing" is an attempt to align the RDN with DNS name associated with the named organisation. The "DC" stands for Domain Component and is an attribute name defined by IETF for use in directory work. Annex A specifies other valid ways to align RDN with DNS as well. Equally valid, the example can choose to align the RDN with the X.500 convention. In such case, the subject string can be "O=com,O=CompanyXYZ,OU=marketing" where O and OU are X.500 standard attributes denoting organisation and organization unit respectively. The alignment choice belongs to the name space designer of each operator. The choice will be reflected in the value of the DN prefix, probably a product configuration parameter. See clause 7 for more information.

Annex D (informative): Interpreting EBNF [13]

This annex provides a very simplified summary of EBNF, and does not modify in any way the reference text in ISO/IEC 14977: "Information technology – Syntactic metalanguage – Extended BNF" [13].

ISO/IEC 14977 [13] specification also provides far greater coverage supported by numerous examples which are not included within this annex.

The EBNF metalanguage is useful for defining rigorous syntax notations and is a notation for defining syntax rules.

The language uses sequences of formal definitions.

Definitions may have several layers of definition. The definitions which are refined are termed as "non terminal symbols".

A term which cannot be defined at a lower level of detail is known as a "terminal symbol". I.e. the "terminal symbols" cannot be further decomposed.

The language permits sentences to be constructed.

The sentences consist of a non terminal, or a terminal symbol, followed by an equality symbol, followed by a formal definition of the symbol.

Each sentence terminates with the semicolon ';' terminal symbol.

Ideally the definitions are read from the top across to the right hand side of the page and downwards.

A definition commences with an identifier (of the thing being defined) followed by an equality sign.

The thing is defined by the symbols and identifiers to the right hand side of the equality symbol, up to the next semicolon ';'.

There is a natural breaking down of definitions, by other definitions until a point is reached that a terminal symbol is reached – which cannot be further defined (e.g. the leaves of definition hierarchy).

There are terminal symbols which permit optional choice, sequence, exclusion, comments to be included in the sentence.

The set of terminal symbols as defined in table 1 of ISO/IEC 14977 [13] are below.

The normal character representing each operator of Extended BNF and its implied precedence is (highest precedence at the top):

'*' repetition-symbol

'-' except-symbol

',' concatenate-symbol

'|' definition-separator-symbol

'=' defining-symbol

';' terminator-symbol

The normal precedence is over-ridden by the following pairs of terminal symbols:

"'"	first-quote-symbol	first-quote-symbol	"'"
'"'	second-quote-symbol	second-quote-symbol	'"'
"(*"	start-comment-symbol	end-comment-symbol	"*)"

'('	start-group-symbol	end-group-symbol	')'
'['	start-option-symbol	end-option-symbol	']'
'{'	start-repeat-symbol	end-repeat-symbol	'}'
'?'	special-sequence-symbol	special-sequence-symbol	'?'

Examples:

```
letter    =   "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L"
            | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "U" | "V" | "W" | "X" | "Y" | "Z" ;

vowel     = "A" | "E" | "I" | "O" | "U" ;  (* a subset of letters *)

consonant = letter – vowel ;  (* the set of letters except vowels *)
```

Annex E (informative): IOC/MOC name recommendation

Recommendation:

3GPP considers the use of many non-alphanumeric characters as valid characters for constructing the IOC names. The Java programming language considers the use of alphanumeric characters plus only two non-alphanumeric characters, i.e. "$" and "_", as valid characters for Java Packages and Java Class names. Because the names of the Java Packages and Java Classes generated by Java programming tools for SS implementation may include MO Class names, a Java environment would have to include a translation mechanism that replaces the invalid characters (if they are used in the IS specification to name an IOC, that is mapped to the same MOC name in a Solution Set) by valid characters. For example, replace "-" by "_". This translation mechanism causes unwanted complexity and reduction in performance of the implementation. Given Java may become popular for coding IRPManager and/or IRPAgent capabilities, this note recommends the specification authors to use valid Java name characters (i.e. all alphanumeric characters plus "$" and "_") to name their IS IOCs and SS MOCs.

Annex F (informative): Change history

Change history								
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Cat**	**Old**	**New**
Jun 2001	SP-12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	--	2.0.0	4.0.0
Dec 2001	SP-14	SP-010641	0001	--	Alignment of Figure C.1 with text in annex C	--	4.0.0	4.1.0
Sep 2001	SP-17	SP-020481	0002	--	Upgrade to Rel-5 (Remove information in the Introduction that is only relevant to Rel-4)	F	4.1.0	5.0.0
Dec 2002	--	--	--	--	Cosmetics	F	5.0.0	5.0.1
Dec 2004	SP-26	SP-040793	0003	--	Correct and convert formal specification from BNF syntax to EBNF with corrections	F	5.0.1	6.0.0
Mar 2005	SP-27	SP-050049	0004	--	Re-introduce text erroneously deleted during implementation of CR 003	F	6.0.0	6.1.0
Jun 2005	SP-28	SP-050285	0005	--	Correct DN string representation formal definition to support wildcard character	F	6.1.0	6.2.0
Dec 2005	SP-30	SP-050726	0006	--	Clarify DN string encoding applicability to all IRP technologies other than CMIP	F	6.2.0	6.3.0
Dec 2005	SP-30	SP-050716	0007	--	Add Annex A from 32.622t	F	6.2.0	6.3.0
Jul 2006	--	--	--	--	Delete duplicated paragraph in the "Introduction" clause	--	6.3.0	6.3.1
Jun 2007	SP-36	--	--	--	Automatic upgrade to Rel-7 (no CR) at freeze of Rel-7. CMIP is still mentioned in spite of discontinuation of CMIP SSs in R7.	--	6.3.1	7.0.0
Sep 2007	SP-37	SP-070612	0008	--	Discontinuing of CMIP Solution Sets in Release 7	F	7.0.0	7.1.0
Dec 2007	SP-38	SP-070731	0009	--	Discontinuation of CMIP Solution Sets in Release 7	F	7.1.0	7.2.0
Dec 2008	SA_42	--	--	--	Upgrade to Release 8	--	7.2.0	8.0.0
Sep 2009	SA_45	SP-090627	0010	--	Cleanup of old references	F	8.0.0	9.0.0
Dec 2009	SA_46	SP-090719	0009	--	Clarify IOC instance naming rule	F	9.0.0	9.1.0
Jun 2010	SA_48	SP-100429	0011	1	Add missing encoding rules for Distinguished Name	F	9.1.0	9.2.0

History

Document history		
V9.1.0	January 2010	Publication
V9.2.0	July 2010	Publication