

**Reemplazada por una versión más reciente**



UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

**UIT-T**

**X.683**

SECTOR DE NORMALIZACIÓN  
DE LAS TELECOMUNICACIONES  
DE LA UIT

(07/94)

**REDES DE DATOS Y COMUNICACIÓN  
ENTRE SISTEMAS ABIERTOS**

**GESTIÓN DE REDES OSI Y ASPECTOS  
DE SISTEMAS – NOTACIÓN DE SINTAXIS  
ABSTRACTA UNO (ASN.1)**

---

**TECNOLOGÍA DE LA INFORMACIÓN –  
NOTACIÓN DE SINTAXIS ABSTRACTA UNO:  
PARAMETRIZACIÓN DE ESPECIFICACIONES  
DE NOTACIÓN DE SINTAXIS  
ABSTRACTA UNO**

**Recomendación UIT-T X.683**

Reemplazada por una versión más reciente

(Anteriormente "Recomendación del CCITT")

---

# Reemplazada por una versión más reciente

## PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. En el UIT-T, que es la entidad que establece normas mundiales (Recomendaciones) sobre las telecomunicaciones, participan unos 179 países miembros, 84 empresas de explotación de telecomunicaciones, 145 organizaciones científicas e industriales y 38 organizaciones internacionales.

Las Recomendaciones las aprueban los Miembros del UIT-T de acuerdo con el procedimiento establecido en la Resolución N.º 1 de la CMNT (Helsinki, 1993). Adicionalmente, la Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, aprueba las Recomendaciones que para ello se le sometan y establece el programa de estudios para el periodo siguiente.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI. El texto de la Recomendación UIT-T X.683 se aprobó el 1 de julio de 1994. Su texto se publica también, en forma idéntica, como Norma Internacional ISO/CEI 8824-4.

---

## NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

© UIT 1996

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

# Reemplazada por una versión más reciente

RECOMENDACIONES UIT-T DE LA SERIE X

## REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

(Febrero de 1994)

### ORGANIZACIÓN DE LAS RECOMENDACIONES DE LA SERIE X

Dominio	Recomendaciones
<b>REDES PÚBLICAS DE DATOS</b>	
Servicios y facilidades	X.1-X.19
Interfaces	X.20-X.49
Transmisión, señalización y conmutación	X.50-X.89
Aspectos de redes	X.90-X.149
Mantenimiento	X.150-X.179
Disposiciones administrativas	X.180-X.199
<b>INTERCONEXIÓN DE SISTEMAS ABIERTOS</b>	
Modelo y notación	X.200-X.209
Definiciones de los servicios	X.210-X.219
Especificaciones de los protocolos en modo conexión	X.220-X.229
Especificaciones de los protocolos en modo sin conexión	X.230-X.239
Formularios para enunciados de conformidad de implementación de protocolo	X.240-X.259
Identificación de protocolos	X.260-X.269
Protocolos de seguridad	X.270-X.279
Objetos gestionados de capa	X.280-X.289
Pruebas de conformidad	X.290-X.299
<b>INTERFUNCIONAMIENTO ENTRE REDES</b>	
Generalidades	X.300-X.349
Sistemas móviles de transmisión de datos	X.350-X.369
Gestión	X.370-X.399
<b>SISTEMAS DE TRATAMIENTO DE MENSAJES</b>	X.400-X.499
<b>DIRECTORIO</b>	X.500-X.599
<b>GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS</b>	
Gestión de redes	X.600-X.649
Denominación, direccionamiento y registro	X.650-X.679
Notación de sintaxis abstracta uno	X.680-X.699
<b>GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS</b>	X.700-X.799
<b>SEGURIDAD</b>	X.800-X.849
<b>APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS</b>	
Cometimiento, concurrencia y recuperación	X.850-X.859
Tratamiento de transacciones	X.860-X.879
Operaciones a distancia	X.880-X.899
<b>TRATAMIENTO ABIERTO DISTRIBUIDO</b>	X.900-X.999



# Reemplazada por una versión más reciente

## ÍNDICE

*Página*

1	Alcance.....	1
2	Referencias normativas .....	1
	2.1 Recomendaciones   Normas Internacionales idénticas.....	1
3	Definiciones .....	1
	3.1 Especificación de notación básica.....	1
	3.2 Especificación de objetos de información.....	1
	3.3 Especificación de constricción.....	1
	3.4 Definiciones adicionales .....	2
4	Abreviaturas .....	2
5	Convenio .....	2
6	Notación .....	2
	6.1 Asignaciones .....	2
	6.2 Definiciones parametrizadas .....	2
	6.3 Símbolos .....	3
7	Ítems de ASN.1 .....	3
8	Asignaciones parametrizadas .....	3
9	Definiciones parametrizadas referentes.....	6
10	Parámetros de la sintaxis abstracta.....	8
	Anexo A – Ejemplos .....	9
	A.1 Ejemplo de utilización de una definición de tipo parametrizado .....	9
	A.2 Ejemplo de utilización de definiciones parametrizadas junto con una clase de objeto de información.....	9
	A.3 Ejemplo de definición finita de un tipo parametrizado .....	10
	A.4 Ejemplo de definición de valor parametrizado .....	11
	A.5 Ejemplo de definición de conjunto de valores parametrizados.....	11
	A.6 Ejemplo de definición de clase parametrizada.....	11
	A.7 Ejemplo de definición de conjunto de objetos parametrizados.....	12
	A.8 Ejemplo de definición de conjunto de objetos parametrizados.....	12
	Anexo B – Resumen de la notación.....	13

# Reemplazada por una versión más reciente

## Sumario

La presente Recomendación | Norma Internacional define las disposiciones para los nombres de referencia parametrizados y asignación parametrizada para tipos de datos, que son útiles a los diseñadores cuando escriben especificaciones en las que algunos aspectos se dejan indefinidos en determinadas etapas del desarrollo para ser completadas en una etapa ulterior con objeto de producir una definición completa de una sintaxis abstracta.

# Reemplazada por una versión más reciente

## Introducción

Los diseñadores de aplicaciones tienen que escribir especificaciones en las cuales ciertos aspectos se dejan sin definir. Dichos aspectos serán definidos posteriormente por uno o más grupos (cada uno a su propia manera), con el fin de producir una especificación completamente definida para utilizarla en la definición de una sintaxis abstracta (una para cada grupo).

En algunos casos, ciertos aspectos de la especificación (por ejemplo, límites) se pueden dejar sin definir, incluso en el momento de la definición de la sintaxis abstracta, siendo completados por la especificación de Perfiles Normalizados Internacionales o perfiles funcionales de algún otro organismo.

NOTA 1 – Un requisito impuesto por esta Recomendación | Norma Internacional es que cualquier aspecto que no esté relacionado únicamente con la aplicación de constricciones tiene que ser completado antes de definir una sintaxis abstracta.

En el caso extremo, algunos aspectos de la especificación se pueden dejar para que el realizador los complete, y se especificarían como parte de la declaración de conformidad de realización del protocolo.

Si bien las disposiciones de la Rec. UIT-T X.681 | ISO/CEI 8824-2 y la Rec. UIT-T X.682 | ISO/CEI 8824-3 proporcionan un marco para la ulterior compleción de partes de una especificación, no resuelven por sí mismas los requisitos mencionados.

Además, a veces un solo diseñador tiene que definir muchos tipos, o muchas clases de objetos de información, o muchos conjuntos de objetos de información, o muchos objetos de información, o muchos valores, que tienen la misma estructura de nivel exterior, pero difieren en los tipos, o clases de objetos de información o conjuntos de objetos de información u objetos de información, o valores, que se utilizan en un nivel interior. En vez de escribir la estructura de nivel exterior para cada uno de estos casos, es útil poder escribirla una vez y dejar que algunas partes sean definidas posteriormente, para hacer referencia a ella y proporcionar la información adicional.

Todos estos requisitos son satisfechos por la provisión de nombres de referencia parametrizados y asignaciones parametrizadas por esta Recomendación | Norma internacional.

La forma sintáctica de un nombre de referencia parametrizado es igual que la del correspondiente nombre de referencia normal, pero son aplicables las siguientes consideraciones adicionales:

- Cuando se asigna en una declaración de asignación parametrizada, va seguida de una lista de nombres de referencia ficticios entre llaves, cada uno acompañado posiblemente por un gobernador; el alcance de estos nombres de referencia está comprendido en el lado derecho de la declaración de asignación, y la propia lista de parámetros.  
NOTA 2 – Esto es lo que hace que sea reconocido como un nombre de referencia parametrizado.
- Cuando se exporta o importa, va seguido de un par de llaves vacías para distinguirlo como un nombre de referencia parametrizado.
- Cuando se utiliza en algún constructivo, va seguido de una lista de construcciones sintácticas, una para cada nombre de referencia ficticio, que proporciona una asignación al nombre de referencia ficticio a los efectos de esa utilización solamente.

Los nombres de referencia ficticios tienen la misma forma sintáctica que el correspondiente nombre de referencia normal, y pueden ser utilizados en cualquier lugar en el lado derecho de la declaración de asignación en que el correspondiente nombre de referencia normal podría ser utilizado. Todas estas utilizaciones tienen que ser coherentes.



**NORMA INTERNACIONAL**

**RECOMENDACIÓN UIT-T**

**TECNOLOGÍA DE LA INFORMACIÓN –  
NOTACIÓN DE SINTAXIS ABSTRACTA UNO:  
PARAMETRIZACIÓN DE ESPECIFICACIONES DE NOTACIÓN  
DE SINTAXIS ABSTRACTA UNO**

**1 Alcance**

La presente Recomendación | Norma Internacional forma parte de la notación de sintaxis abstracta uno (ASN.1) y define la notación para la parametrización de especificaciones de ASN.1.

**2 Referencias normativas**

Las siguientes Recomendaciones y Normas Internacionales contienen disposiciones que, al hacerse referencia a las mismas en este texto, constituyen disposiciones de la presente Recomendación | Norma Internacional. En el momento de su publicación, las ediciones indicadas eran válidas. Todas las Recomendaciones y Normas pueden ser objeto de revisión, y se alienta a las partes en acuerdos basados en esta Recomendación | Norma Internacional a investigar la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y Normas indicadas. Los miembros de la CEI y la ISO mantienen registros de las Normas Internacionales en vigor. La Oficina de Normalización de las Telecomunicaciones de la UIT mantienen una lista de las Recomendaciones UIT-T actualmente válidas.

**2.1 Recomendaciones | Normas Internacionales idénticas**

- Recomendación UIT-T X.680 (1994) | ISO/CEI 8824-1:1994, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica.*
- Recomendación UIT-T X.681 (1994) | ISO/CEI 8824-2:1994, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de los objetos informacionales.*
- Recomendación UIT-T X.682 (1994) | ISO/CEI 8824-3:1994, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.*

**3 Definiciones**

A los efectos de la presente Recomendación | Norma Internacional, se aplican las siguientes definiciones.

**3.1 Especificación de notación básica**

Esta Recomendación | Norma Internacional utiliza los términos definidos en la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**3.2 Especificación de objetos de información**

Esta Recomendación | Norma Internacional utiliza los términos definidos en la Rec. UIT-T X.681 | ISO/CEI 8824-2.

**3.3 Especificación de constricción**

Esta Recomendación | Norma Internacional utiliza los términos definidos en la Rec. UIT-T X.682 | ISO/CEI 8824-3.

### 3.4 Definiciones adicionales

**3.4.1 nombre de referencia normal:** Nombre de referencia definido, sin parámetros, por medio de una «Asignación» distinta a una «Asignación parametrizada». Este nombre hace referencia a una definición completa y no se suministra con parámetros reales cuando se utiliza.

**3.4.2 nombre de referencia parametrizado:** Nombre de referencia que se define utilizando una asignación parametrizada, que hace referencia a una definición incompleta y que, por tanto, se debe suministrar con parámetros reales cuando se utiliza.

**3.4.3 tipo parametrizado:** Tipo definido utilizando una asignación de tipo parametrizada y cuyos componentes son definiciones incompletas que se debe suministrar con parámetros reales cuando se utiliza el tipo.

**3.4.4 valor parametrizado:** Valor definido utilizando una asignación de valor parametrizada y cuyo valor se especifica de manera incompleta, por lo que se debe suministrar con parámetros reales cuando se utiliza.

**3.4.5 conjunto de valores parametrizados:** Conjunto de valores definido utilizando una asignación de conjunto de valores parametrizada y cuyos valores se especifican de manera incompleta, por lo que se debe suministrar con parámetros reales cuando se utiliza.

**3.4.6 clase de objeto parametrizado:** Clase de objeto de información definida utilizando una asignación de clase de objeto parametrizada y cuyas especificaciones de campo se especifican de manera incompleta, por lo que se debe suministrar con parámetros reales cuando se utiliza.

**3.4.7 objeto parametrizado:** Objeto de información definido utilizando una asignación de objeto parametrizada y cuyos componentes se especifican de manera incompleta, por lo que se debe suministrar con parámetros reales cuando se utiliza.

**3.4.8 conjunto de objetos parametrizados:** Conjunto de objetos de información definido utilizando una asignación de conjunto de objetos parametrizada y cuyos objetos se especifican de manera incompleta, por lo que se debe suministrar con parámetros reales cuando se utiliza.

**3.4.9 constricción variable:** Constricción empleada al especificar una sintaxis abstracta parametrizada y que depende de algún parámetro de la sintaxis abstracta.

## 4 Abreviaturas

ASN.1 Notación de sintaxis abstracta uno (*abstract syntax notation one*).

## 5 Convenio

Esta Recomendación | Norma Internacional emplea el convenio de notación definido en la Rec. UIT-T X.680 | ISO/CEI 8824-1, cláusula 5.

## 6 Notación

Esta cláusula resume la notación definida en esta Recomendación | Norma Internacional.

### 6.1 Asignaciones

En esta Recomendación | Norma Internacional se define la siguiente notación que se puede utilizar como una alternativa "Assignment" (asignación) (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1, cláusula 10):

- ParameterizedAssignment (asignación parametrizada) (véase 8.1).

### 6.2 Definiciones parametrizadas

**6.2.1** La siguiente notación que se puede utilizar como una alternativa a "DefinedType" (tipo definido) (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1, subcláusula 11.1) se define en esta Recomendación | Norma Internacional:

- ParameterizedType (tipo parametrizado) (véase 9.2).

**6.2.2** La siguiente definición que se puede utilizar como una alternativa a "DefinedValue" (valor definido) (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1, subcláusula 11.1) se define en esta Recomendación | Norma Internacional:

- ParameterizedValue (valor parametrizado) (véase 9.2).

**6.2.3** La siguiente notación que se puede utilizar como una alternativa a "DefinedType" (tipo definido) (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1, subcláusula 11.1) se define en esta Recomendación | Norma Internacional:

- ParameterizedValueType (tipo conjunto de valores parametrizados) (véase 9.2).

**6.2.4** La siguiente notación que se puede utilizar como una alternativa a "ObjectClass" (clase de objeto) (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, subcláusula 9.2) se define en esta Recomendación | Norma Internacional:

- ParameterizedObjectClass (clase de objeto parametrizado) (véase 9.2).

**6.2.5** La siguiente notación que se puede utilizar como una alternativa a "Object" (objeto) (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, subcláusula 11.2) se define en esta Recomendación | Norma Internacional:

- ParameterizedObject (objeto parametrizado) (véase 9.2).

**6.2.6** La siguiente notación que se puede utilizar como una alternativa a "ObjectSet" (conjunto de objetos) (véase la Rec. UIT-T X.681 | ISO/CEI 8824-2, subcláusula 12.2) se define en esta Recomendación | Norma Internacional:

- ParameterizedObjectSet (conjunto de objetos parametrizados) (véase 9.2).

## 6.3 Símbolos

La siguiente notación que se puede utilizar como una alternativa a "Símbolo" (véase la Rec. UIT-T X.680 | ISO/CEI 8824-1, subcláusula 10.1) se define en esta Recomendación | Norma Internacional.

- ParameterizedReference (Referencia parametrizada) (véase 9.1).

## 7 Ítems de ASN.1

En esta Recomendación | Norma Internacional se utilizan los ítems de ASN.1 especificados en la Rec. UIT-T X.680 | ISO/CEI 8824-1, cláusula 9.

## 8 Asignaciones parametrizadas

**8.1** Hay declaraciones de asignaciones parametrizadas que corresponden a cada una de las declaraciones de asignaciones especificadas en la Rec. UIT-T X.680 | ISO/CEI 8824-1 y Rec. UIT-T X.681 | ISO/CEI 8824-2. El constructivo "ParameterizedAssignment" es:

```

ParameterizedAssignment ::=
    ParameterizedTypeAssignment |
    ParameterizedValueAssignment |
    ParameterizedValueTypeAssignment |
    ParameterizedObjectClassAssignment |
    ParameterizedObjectAssignment |
    ParameterizedObjectSetAssignment
    
```

**8.2** Cada "Parameterized<X>Assignment" (asignación X parametrizada) tiene la misma sintaxis que "<X>Assignment" (asignación X) salvo que después del ítem inicial hay una "ParameterList" (lista de parámetros). De este modo, el ítem inicial se convierte en un nombre de referencia parametrizado (véase 3.4.2):

NOTA – La Rec. UIT-T.680 | ISO/CEI 8824-1 impone el requisito de que todos los nombres de referencia asignados dentro de un módulo, sean parametrizados o no, deben ser distintos.

```

ParameterizedTypeAssignment ::=
    typereference
    ParameterList
    "::~="
    Type
    
```

```

ParameterizedValueAssignment ::=
    valuereference
    ParameterList
    Type
    "::~="
    Value
    
```

**ParameterizedValueSetTypeAssignment ::=**

**typereference**  
**ParameterList**  
**Type**  
**"::="**  
**ValueSet**

**ParameterizedObjectClassAssignment ::=**

**objectclassreference**  
**ParameterList**  
**"::="**  
**ObjectClass**

**ParameterizedObjectAssignment ::=**

**objectreference**  
**ParameterList**  
**DefinedObjectClass**  
**"::="**  
**Object**

**ParameterizedObjectSetAssignment ::=**

**objectsetreference**  
**ParameterList**  
**DefinedObjectClass**  
**"::="**  
**ObjectSet**

**8.3** "ParameterList" es una lista de "Parámetros" entre llaves.

**ParameterList ::= "{" Parameter "," + "}"**

Cada "Parámetro" consiste en una "DummyReference" (referencia ficticia) y posiblemente un "ParamGovernor" (gobernador de parámetro).

**Parameter ::= ParamGovernor ":" DummyReference | DummyReference**

**ParamGovernor ::= Governor | DummyGovernor**

**Governor ::= Type | DefinedObjectClass**

**DummyGovernor ::= DummyReference**

**DummyReference ::= Reference**

Una "DummyReference" en "Parameter" puede representar:

- a) un "Type" o una "DefinedObjectClass", en cuyo caso no habrá "ParamGovernor";
- b) un "Value" o un "ValueSet", en cuyo caso estará presente el "ParamGovernor": si el "ParamGovernor" es un "Governor" será un "Type" y si el "ParamGovernor" es un "DummyGovernor" el parámetro real del "ParamGovernor" será un "Type";
- c) un "Object" o un "ObjectSet", en cuyo caso estará presente el "ParamGovernor": si el "ParamGovernor" es un "Governor" será una "DefinedObjectClass" y si el "ParamGovernor" es un "DummyGovernor" el parámetro real del "ParamGovernor" será una "DefinedObjectClass".

Un "DummyGovernor" será una "DummyReference" que no tiene "Governor".

**8.4** El alcance de una "DummyReference" que aparece en una "ParameterList" es la propia "ParameterList", junto con esa parte de "ParameterizedAssignment", que sigue a "::=". "DummyReference" oculta cualquier otra "Reference" con la misma ortografía en ese ámbito.

**8.5** La utilización de "DummyReference" dentro de su ámbito concordará con su forma sintáctica y, cuando sea aplicable, Governor, y todos los usos de la "DummyReference" concordarán entre sí.

NOTA – Cuando la forma sintáctica de un nombre de referencia ficticio (por ejemplo, si es una "objectclassreference" o "typereference") es ambigua, la ambigüedad se puede resolver normalmente en la primera utilización del nombre de referencia ficticio en el lado derecho de la declaración de asignación. En lo sucesivo, se conoce la naturaleza del nombre de referencia ficticio. Sin embargo, la naturaleza de la referencia ficticia no está terminada solamente por el lado derecho de la declaración de asignación cuando a su vez se utiliza solamente como un parámetro real en una referencia parametrizada; en este caso, la naturaleza de la referencia ficticia se debe determinar examinando la definición de esta referencia parametrizada. Se advierte a los usuarios de la notación que esta práctica puede hacer que las especificaciones de ASN.1 sean menos claras y se aconseja que se proporcionen los comentarios adecuados para explicar esto a los lectores humanos.

### Ejemplo

Considérese la siguiente asignación de clase objeto parametrizada:

```
PARAMETERIZED-OBJECT-CLASS { TypeParam, INTEGER:valueParam, INTEGER:ValueSetParam } ::=
    CLASS {
        &valueField1      TypeParam,
        &valueField2      INTEGER DEFAULT valueParam,
        &valueField3      INTEGER (ValueSetParam),
        &ValueSetField    INTEGER DEFAULT { ValueSetParam }
    }
```

Para determinar la utilización apropiada de "DummyReference" en el ámbito de "ParameterizedAssignment", y sólo con ese fin, se puede considerar que "DummyReference" se define como sigue:

```
TypeParam ::= UnspecifiedType
valueParam INTEGER ::= unspecifiedIntegerValue
ValueSetParam INTEGER ::= { UnspecifiedIntegerValueSet }
```

donde:

- a) TypeParam es una "DummyReference" que representa un "Tipo". Por tanto, TypeParam se puede utilizar cuando se puede utilizar "typereference", por ejemplo, como un "Tipo" para el campo de valor de tipo fijo valueField1.
- b) valueParam es una "DummyReference" que representa un valor de un tipo entero. Por tanto, valueParam se puede utilizar cuando se puede utilizar "valuereference" de un valor entero; por ejemplo, como un valor por defecto para el campo de valor de tipo fijo valueField2.
- c) ValueSetParam es una "DummyReference" que representa un conjunto de valores de un tipo entero. Por tanto, ValueSetParam se puede utilizar cuando se puede utilizar "typereference" de un valor entero, por ejemplo, como un "Tipo" en la notación "ContainedSubtype" para valueField3 y ValueSetField.

**8.6** Cada "DummyReference" se empleará como mínimo una vez dentro de su ámbito.

NOTA – Si "DummyReference" no ha aparecido, el correspondiente "ActualParameter" no tendrá efecto sobre la definición y sencillamente sería "descartado", mientras que al usuario pudiera parecerle que alguna especificación tomó el lugar.

Las "ParameterizedValueAssignment", "ParameterizedValueSetTypeAssignment", "ParameterizedObjectAssignment" y "ParameterizedObjectSetAssignment" que contienen una referencia directa o indirecta a ellas mismas son ilegales.

**8.7** En la definición de un "ParameterizedType", un "ParameterizedValueSet" o una "ParameterizedObjectClass" no se pasará "DummyReference" como un tipo rotulado (como un parámetro real) a una referencia recursiva a ese "ParameterizedType", ese "ParameterizedValueSet" o esa "ParameterizedObjectClass" (véase A.3).

**8.8** En la definición de un "ParameterizedType", un "ParameterizedValueSet", o una "ParameterizedObjectClass", no se efectuará una referencia circular al ítem que se define a menos que esa referencia esté marcada directa o indirectamente OPTIONAL o en el caso de "ParameterizedType" y "ParameterizedValueSet", efectuado a través de una referencia a un Tipo elección, al menos una de cuyas alternativas es no circular en la definición.

**8.9** El gobernador de una "DummyReference" (referencia ficticia) no incluirá una referencia a otra "DummyReference" si esa otra "DummyReference" también tiene un gobernador.

- 8.10 En una asignación parametrizada, el lado derecho de "::=" no consistirá solamente en una "DummyReference".
- 8.11 El gobernador de una "DummyReference" no necesitará el conocimiento de la "DummyReference" ni del nombre de la referencia parametrizada que se define.

## 9 Definiciones parametrizadas referentes

9.1 Dentro de una "SymbolList" (lista de símbolos) [en "Exports" (exportaciones) o "Imports" (importaciones)], una definición parametrizada será referenciada por una "ParameterizedReference".

**ParameterizedReference ::= Reference | Reference "{" "}"**

donde "Reference" es el primer ítem en "ParameterizedAssignment", como se especifica en 8.2.

NOTA – La primera alternativa de una "ParameterizedReference" únicamente se proporciona como una ayuda para la comprensión humana. Ambas alternativas tienen el mismo significado.

9.2 De manera distinta a como se hace en "Exports" o "Imports", una definición parametrizada será referenciada por un constructivo "Parameterized<X>", que se puede utilizar como una alternativa al "<X>" correspondiente.

**ParameterizedType ::=**  
**SimpleDefinedType**  
**ActualParameterList**

**SimpleDefinedType ::=**  
**Externaltypereference |**  
**typereference**

**ParameterizedValue ::=**  
**SimpleDefinedValue**  
**ActualParameterList**

**SimpleDefinedValue ::=**  
**Externalvaluereference |**  
**valuereference**

**ParameterizedValueSetType ::=**  
**SimpleDefinedType**  
**ActualParameterList**

**ParameterizedObjectClass ::=**  
**DefinedObjectClass**  
**ActualParameterList**

**ParameterizedObjectSet ::=**  
**DefinedObjectSet**  
**ActualParameterList**

**ParameterizedObject ::=**  
**DefinedObject**  
**ActualParameterList**

9.3 El nombre de referencia en "Defined<X>" será un nombre de referencia asignado en una "ParameterizedAssignment".

9.4 Las restricciones impuestas a la alternativa "Defined<X>" que ha de utilizarse y que se especifican en la Rec. UIT-T X.680 | ISO/CEI 8824-1 y Rec. UIT-T X.681 | ISO/CEI 8824-2 para los nombres de referencia normales, se aplican igualmente a los correspondientes nombres de referencia parametrizados.

NOTA – En esencia, las restricciones son las siguientes. Cada "Defined<X>" tiene dos alternativas, "<x>reference" y "External<x>reference". La primera se utiliza dentro del módulo de definición o si la definición ha sido importada y cuando no hay conflicto de nombres; la segunda se utiliza cuando no hay importaciones enumeradas (desaconsejado), o si hay un conflicto de nombres entre una importación y una definición local (también desaconsejado) o entre importaciones.

9.5 "ActualParameterList" es:

**ActualParameterList ::=**  
**"{" ActualParameter "," + "}"**

**ActualParameter ::=**  
**Type |**  
**Value |**

```

ValueSet      |
DefinedObjectClass |
Object        |
ObjectSet

```

**9.6** Habrá exactamente un "ActualParameter" para cada "Parameter" en la correspondiente "ParameterizedAssignment" y aparecerán en el mismo orden. La elección particular de "ActualParameter" y el gobernador (si lo hubiere) se determinará mediante examen de la forma sintáctica de "Parameter" y el entorno en el cual se produce en la "ParameterizedAssignment". La forma de "ActualParameter" será la forma requerida para colocar la "DummyReference" dondequiera en su ámbito (véase 8.4).

### Ejemplo

La definición de clase de objeto parametrizado del ejemplo anterior (véase 8.5) se puede referenciar, por ejemplo, como sigue:

```

MY-OBJECT-CLASS ::= PARAMETERIZED-OBJECT-CLASS { BIT STRING, 123, {4 | 5 | 6} }

```

**9.7** El parámetro real toma el lugar del nombre de referencia ficticio al determinar el tipo, valor, conjunto de valores, clase de objeto, objeto o conjunto de objetos reales que está siendo referenciado por esta utilización del nombre de referencia parametrizado.

**9.8** El significado de cualquier referencia, que aparezca en "ActualParameter" y el rótulo por defecto aplicable a cualesquiera rótulos que así aparecen, se determina de acuerdo con el entorno de "ActualParameter" en vez del entorno de la correspondiente "DummyReference".

NOTA – De este modo, la parametrización, como la referenciación, tipo de selección y "COMPONENT OF", entre otros, no es exactamente una sustitución textual.

### Ejemplo

Considérense los siguientes módulos:

```

M1 DEFINITIONS AUTOMATIC TAGS ::= BEGIN
  EXPORTS T1;

  T1 ::= SET {
    f1    INTEGER,
    f2    BOOLEAN
  }
END

M2 DEFINITIONS EXPLICIT TAGS ::= BEGIN
  IMPORTS T1 FROM M1;

  T3 ::= T2{T1}

  T2{X} ::= SEQUENCE {
    a    INTEGER,
    b    X
  }
END

```

La aplicación de 9.8 entraña que el rótulo del componente f1 de T3 (es decir @T3.b.f1) será rotulado implícitamente porque el entorno de rotulación del parámetro ficticio X, a saber, rotulación explícita, no afecta a la rotulación de los componentes del parámetro real T1.

Considérense el módulo M3.

```

M3 DEFINITIONS AUTOMATIC TAGS ::= BEGIN
  IMPORTS T1 FROM M1;

  T5 ::= T4{T1}

  T4{Y} ::= SEQUENCE {
    a    INTEGER,
    b    Y
  }
END

```

La aplicación de la subcláusula 28.6 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, entraña que el rótulo del componente b de T5 (es decir, @T5.b) será rotulado explícitamente porque el parámetro ficticio (Y) se rotula siempre explícitamente, con lo que @T5 es equivalente a:

```
T5 ::= SEQUENCE {
    a  [0] IMPLICIT INTEGER,
    b  [1] EXPLICIT SET {
        f1  [0] INTEGER,
        f2  [1] BOOLEAN
    }
}
```

mientras que @T3 es equivalente a:

```
T3 ::= SEQUENCE {
    a  INTEGER,
    b  SET {
        f1  [0] IMPLICIT INTEGER,
        f2  [1] IMPLICIT BOOLEAN
    }
}
```

## 10 Parámetros de la sintaxis abstracta

**10.1** El Anexo B a la Rec. UIT-T X.681 | ISO/CEI 8824-2 proporciona la clase de objeto de información ABSTRACT-SYNTAX (sintaxis abstracta) y recomienda su uso para definir sintaxis abstractas, utilizando como ejemplo una sintaxis abstracta definida como el conjunto de valores de un solo tipo ASN.1 que no estaba parametrizado en el nivel externo.

**10.2** Si el tipo ASN.1 utilizado para definir la sintaxis abstracta *está* parametrizado, se pueden suministrar algunos parámetros como parámetros reales cuando se define la sintaxis abstracta, mientras que otros pueden dejarse como parámetros de la propia sintaxis abstracta.

### Ejemplo

Si se ha definido un tipo parametrizado denominado YYY-PDU con dos referencias ficticias (la primera, digamos, un conjunto de objetos de alguna clase de objeto definida, y la segunda un valor entero para una vinculación), entonces:

```
yyy-Abstract-Syntax { INTEGER:bound } ABSTRACT-SYNTAX ::=
    { YYY-PDU { {ValidObjects} , bound } IDENTIFIED BY {yyy 5} }
```

define una sintaxis abstracta parametrizada en la cual el conjunto de objetos ha sido resuelto, pero "bound" (vinculación) permanece como un parámetro de la sintaxis abstracta.

Un parámetro de sintaxis abstracta se utilizará:

- directa o indirectamente en el contexto de una restricción;
- directa o indirectamente como parámetro real que, en su momento, se utiliza en el contexto de una restricción.

NOTA – Véase el ejemplo de A.2 y el ejemplo de G.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

**10.3** Una restricción cuyo conjunto de valores depende de uno o más parámetros de la sintaxis abstracta es una restricción variable. Estas restricciones se determinan después de la definición de la sintaxis abstracta (tal vez mediante perfiles normalizados internacionales o en enunciados de conformidad de realización de protocolo).

NOTA – Si en alguna parte de la cadena de definiciones que participan en la especificación de los valores de restricciones aparece un parámetro de la sintaxis abstracta, la restricción es una restricción variable. Es una restricción variable incluso si el conjunto de valores de la restricción resultante es independiente del valor real del parámetro de la sintaxis abstracta.

**Ejemplo** – El valor de (((1..3) EXCEPT a) UNION (1..3)) es siempre 1..3 no importa cuál sea el valor de "a", y no obstante, es aún una restricción variable si "a" es un parámetro de la sintaxis abstracta.

**10.4** Formalmente, una restricción variable no restringe al conjunto de valores en la sintaxis abstracta.

NOTA – Se recomienda decididamente que las restricciones que se prevé permanezcan como restricciones variables en una sintaxis abstracta tengan una especificación de excepción, utilizando la notación proporcionada por la Rec. UIT-T X.680 | ISO/CEI 8824-1, subcláusula 43.4.

## Anexo A

### Ejemplos

(Este anexo no forma parte integrante de la presente Recomendación | Norma Internacional)

#### A.1 Ejemplo de utilización de una definición de tipo parametrizado

Supongamos que un diseñador de protocolo necesita frecuentemente transportar un autenticador (authenticator) con uno o más de los campos del protocolo. Este será transportado como una BIT STRING (cadena de bits) a lo largo del campo. Sin parametrización, el «autenticador» tendría que ser definido como una BIT STRING, y tendría que ser añadido siempre que tuviera que aparecer, con un texto para identificar a qué se aplica. Como otra posibilidad, el diseñador podría adoptar el método de convertir cualquier campo que tiene un autenticador en una SEQUENCE (secuencia) de ese campo y autenticador. El mecanismo de parametrización proporciona una manera abreviada conveniente de realizar esta tarea.

En primer lugar, se define el tipo parametrizado SIGNED { }:

```
SIGNED { ToBeSigned } ::= SEQUENCE
{
    authenticated-data  ToBeSigned,
    authenticator       BIT STRING
}
```

después, en el cuerpo del protocolo, la notación (por ejemplo)

```
SIGNED { OrderInformation }
```

es una notación de tipo que representa

```
SEQUENCE
{
    authenticated-data  OrderInformation,
    authenticator       BIT STRING
}
```

Supongamos además que para algunos campos, el emisor debe tener la opción de añadir o no el autenticador. Esto se puede lograr haciendo BIT STRING facultativa, pero una solución más elegante (menos bits en la línea) sería definir otro tipo parametrizado:

```
OPTIONALLY-SIGNED {ToBeSigned} ::= CHOICE
{
    unsigned-data      [0]  ToBeSigned,
    signed-data        [1]  SIGNED { ToBeSigned }
}
```

NOTA – La rotulación en CHOICE no es necesaria si el escritor garantiza que ninguno de los usos del tipo parametrizado produce un argumento real que sea BIT STRING (el tipo de SIGNED), pero es útil evitar errores en otros lugares de la especificación.

#### A.2 Ejemplo de utilización de definiciones parametrizadas junto con una clase de objeto de información

Se utilizarán clases de objetos de información para acoplar todos los parámetros de una sintaxis abstracta. De esta manera, el número de parámetros de una sintaxis abstracta puede reducirse a uno que es un ejemplar de la clase de colección. La producción "InformationFromObject" (información procedente de objeto) puede utilizarse para extraer información del objeto del parámetro.

##### Ejemplo

```
-- Un ejemplar de esta clase contiene todos los parámetros de la sintaxis abstracta
-- Message-PDU.
```

```
MESSAGE-PARAMETERS ::= CLASS {
    &maximum-priority-level  INTEGER,
    &maximum-message-buffer-size  INTEGER,
    &maximum-reference-buffer-size  INTEGER
}
```

```

WITH SYNTAX {
    THE MAXIMUM PRIORITY LEVEL IS          &maximum-priority-level
    THE MAXIMUM MESSAGE BUFFER SIZE IS     &maximum-message-buffer-size
    THE MAXIMUM REFERENCE BUFFER SIZE IS   &maximum-reference-buffer-size
}

```

-- La producción "ValueFromObject" (valor procedente de objeto) se utiliza para extraer  
-- valores del parámetro de sintaxis abstracta, "param". Los valores pueden utilizarse  
-- solamente en constricciones. Además, el parámetro se pasa a otro tipo parametrizado.

```

Message-PDU { MESSAGE-PARAMETERS : param } ::= SEQUENCE {
    priority-level  INTEGER (0..param.&maximum-priority-level),
    message        BMPString (SIZE (0..param.&maximum-message-buffer-size)),
    reference      Reference { param }
}

```

```

Reference { MESSAGE-PARAMETERS : param } ::=
    SEQUENCE OF
        IA5String (SIZE (0..param.&maximum-reference-buffer-size))

```

-- Definición de un objeto de información de sintaxis abstracta parametrizada.  
-- El parámetro de la sintaxis abstracta se utiliza solamente en constricciones.

```

message-Abstract-Syntax { MESSAGE-PARAMETERS : param }
ABSTRACT-SYNTAX ::=
{
    Message-PDU { param }
    IDENTIFIED BY { joint-iso-ccitt asn1(1) examples(123) 0 }
}

```

La clase MESSAGE-PARAMETERS y el objeto de sintaxis abstracta parametrizada, message-Abstract-Syntax, se utilizan de la siguiente manera:

-- Este ejemplar de MESSAGE-PARAMETERS define valores de parámetros de la  
-- sintaxis abstracta.

```

my-message-parameters MESSAGE-PARAMETERS ::= {
    THE MAXIMUM PRIORITY LEVEL IS 10
    THE MAXIMUM MESSAGE BUFFER SIZE IS 2000
    THE MAXIMUM REFERENCE BUFFER SIZE IS 100
}

```

-- La sintaxis abstracta puede definirse ahora con todas las constricciones de variables especificadas.

```

my-message-Abstract-Syntax ABSTRACT-SYNTAX ::=
    message-Abstract-Syntax { my-message-parameters }

```

### A.3 Ejemplo de definición finita de un tipo parametrizado

Cuando se especifique un tipo parametrizado que represente una lista genérica, se hará de modo que la notación ASN.1 resultante sea finita. Por ejemplo, se puede especificar:

```

List1 { ElementTypeParam } ::= SEQUENCE {
    elem    ElementTypeParam,
    next    List1 { ElementTypeParam } OPTIONAL
}

```

que es finita, porque cuando se utiliza,

```

IntegerList1 ::= List1 { INTEGER }

```

la notación ASN.1 resultante es tal como usted la definiría normalmente:

```

IntegerList1 ::= SEQUENCE {
    elem    INTEGER,
    next    IntegerList1 OPTIONAL
}

```

Compárese con lo siguiente:

```
List2 { ElementTypeParam } ::= SEQUENCE {
    elem    ElementTypeParam,
    next    List2 { [0] ElementTypeParam } OPTIONAL
}

IntegerList2 ::= List2 { INTEGER }
```

en donde la notación ASN.1 es infinita:

```
IntegerList2 ::= SEQUENCE {
    elem    INTEGER,
    next    SEQUENCE {
        elem    [0] INTEGER,
        next    SEQUENCE {
            elem    [0][0] INTEGER,
            next    SEQUENCE {
                elem    [0][0][0] INTEGER,
                next    SEQUENCE {
                    ... -- y así sucesivamente
                } OPTIONAL
            } OPTIONAL
        } OPTIONAL
    } OPTIONAL
}
```

#### A.4 Ejemplo de definición de valor parametrizado

Si un valor de cadena parametrizado se define como a continuación se indica:

```
genericBirthdayGreeting { IA5String : name } IA5String ::= { "Happy birthday, ", name, "!!" }
```

los dos valores de cadena siguientes son iguales:

```
greeting1 IA5String ::= genericBirthdayGreeting { "John" }
greeting2 IA5String ::= "Happy birthday, John!!"
```

#### A.5 Ejemplo de definición de conjunto de valores parametrizados

Si dos conjuntos de valores parametrizados se definen como a continuación se indica:

```
QuestList1 { IA5String : extraQuest } IA5String ::= { "Jack" | "John" | extraQuest }
QuestList2 { IA5String : ExtraQuests } IA5String ::= { "Jack" | "John" | ExtraQuests }
```

los conjuntos de valores siguientes denotan el mismo conjunto de valores:

```
SetOfQuests1 IA5String ::= { QuestList1 { "Jill" } }
SetOfQuests2 IA5String ::= { QuestList2 { { "Jill" } } }
SetOfQuests3 IA5String ::= { "Jack" | "John" | "Jill" }
```

y los conjuntos de valores siguientes denotan el mismo conjunto de valores:

```
SetOfQuests4 IA5String ::= { QuestList2 { { "Jill" | "Mary" } } }
SetOfQuests5 IA5String ::= { "Jack" | "John" | "Jill" | "Mary" }
```

Obsérvese que un conjunto de valores se especifica *siempre* entre llaves, incluso cuando es una referencia de conjunto de valores parametrizados. Si se omiten las llaves de una referencia a un "identifier" (identificador), que se creó en una asignación de conjunto de valores o de una referencia a un "ParameterizedValueSetType", la notación es la de un "Type", no la de un conjunto de valores.

#### A.6 Ejemplo de definición de clase parametrizada

La siguiente clase parametrizada puede utilizarse para definir clases de error que contienen códigos de error de tipos diferentes. Se señala que el parámetro "ErrorCodeType" (tipo de código de error) se utiliza solamente como un "DummyGovernor" (gobernador ficticio) para el parámetro "ValidErrorCodes" (códigos de error válidos).

```
GENERIC-ERROR { ErrorCodeType, ErrorCodeType : ValidErrorCodes } ::= CLASS {
    &errorCode    ValidErrorCodes
}
```

```
WITH SYNTAX {
  CODE &errorCode
}
```

La definición de clase parametrizada puede utilizarse como a continuación se indica para definir clases diferentes que comparten algunas características, como la misma sintaxis definida.

```
ERROR-1 ::= GENERIC-ERROR { INTEGER, { 1 | 2 | 3 } }
ERROR-2 ::= GENERIC-ERROR { ErrorCodeString, { StringErrorCodes } }
ERROR-3 ::= GENERIC-ERROR { EnumeratedErrorCode, { fatal | error } }
ErrorCodeString ::= IA5String (SIZE (4))
StringErrorCodes ErrorCodeString ::= { "E001" | "E002" | "E003" }
EnumeratedErrorCode ::= ENUMERATED { fatal, error, warning }
```

Las clases definidas pueden utilizarse entonces como sigue:

```
My-Errors ERROR-2 ::= { { CODE "E001" } | { CODE "E002" } }
fatalError ERROR-3 ::= { CODE fatal }
```

### A.7 Ejemplo de definición de conjunto de objetos parametrizados

La definición del conjunto de objetos parametrizados AllTypes forma un conjunto de objetos que contiene un conjunto básico de objetos, BaseTypes, y un conjunto de objetos adicionales que son suministrados como un parámetro, AdditionalTypes.

```
AllTypes { TYPE-IDENTIFIER : AdditionalTypes } TYPE-IDENTIFIER ::= { BaseTypes | AdditionalTypes }
BaseTypes TYPE-IDENTIFIER ::= {
  { BaseType-1 IDENTIFIED BY basic-type-obj-id-value-1 } |
  { BaseType-2 IDENTIFIED BY basic-type-obj-id-value-2 } |
  { BaseType-3 IDENTIFIED BY basic-type-obj-id-value-3 }
}
```

La definición del conjunto de objetos parametrizados, AllTypes, puede utilizarse como sigue:

```
My-All-Types TYPE-IDENTIFIER ::= { AllTypes {
  { My-Type-1 IDENTIFIED BY my-obj-id-value-1 } |
  { My-Type-2 IDENTIFIED BY my-obj-id-value-2 } |
  { My-Type-3 IDENTIFIED BY my-obj-id-value-3 }
} }
```

### A.8 Ejemplo de definición de conjunto de objetos parametrizados

El tipo definido en la cláusula A.4 de la Rec. UIT-T X.682 | ISO/CEI 8824-3 puede utilizarse en una definición de sintaxis abstracta parametrizada de la siguiente manera:

```
-- PossibleBodyTypes es un parámetro de una sintaxis abstracta.
message-abstract-syntax { MHS-BODY-CLASS : PossibleBodyTypes } ABSTRACT-SYNTAX ::= {
  INSTANCE OF MHS-BODY-CLASS ({PossibleBodyTypes})
  IDENTIFIED BY { joint-iso-itu asn1(1) examples(1) 123 }
}

-- Este conjunto de objetos enumera todos los pares posibles de valores e
-- identificadores de tipo del tipo ejemplar de. El conjunto de objetos se utiliza como un
-- parámetro real para la definición de la sintaxis abstracta parametrizada.
My-Body-Types MHS-BODY-CLASS ::= {
  { My-First-Type IDENTIFIED BY my-first-obj-id } |
  { My-Second-Type IDENTIFIED BY my-second-obj-id }
}
my-message-abstract-syntax ABSTRACT-SYNTAX ::=
  message-abstract-syntax { { My-Body-Types } }
```

**Anexo B****Resumen de la notación**

(Este anexo no forma parte integrante de la presente Recomendación | Norma Internacional)

Los siguientes ítems se definen en la Rec. UIT-T X.680 | ISO/CEI 8824-1 y se utilizan en esta Recomendación | Norma Internacional:

**typereference**  
**valuereference**  
 "::<=" "  
 "{" "  
 "}" "  
 "," "

Los siguientes ítems se definen en la Rec. UIT-T X.681 | ISO/CEI 8824-2 y se utilizan en esta Recomendación | Norma Internacional.

**objectclassreference**  
**objectreference**  
**objectsetreference**

Las siguientes producciones se definen en la Rec. UIT-T X.680 | ISO/CEI 8824-1 y se utilizan en esta Recomendación | Norma Internacional.

**DefinedType**  
**DefinedValue**  
**Reference**  
**Type**  
**Value**  
**ValueSet**

Las siguientes producciones se definen en la Rec. UIT-T X.681 | ISO/CEI 8824-2 y se utilizan en esta Recomendación | Norma Internacional.

**DefinedObjectClass**  
**DefinedObject**  
**DefinedObjectSet**  
**ObjectClass**  
**Object**  
**ObjectSet**

Las siguientes producciones se definen en la presente Recomendación | Norma Internacional.

**ParameterizedAssignment ::=**  
 ParameterizedTypeAssignment |  
 ParameterizedValueAssignment |  
 ParameterizedValueSetTypeAssignment |  
 ParameterizedObjectClassAssignment |  
 ParameterizedObjectAssignment |  
 ParameterizedObjectSetAssignment

**ParameterizedTypeAssignment ::=**  
 typereference ParameterList "::<=" Type

**ParameterizedValueAssignment ::=**  
 valuereference ParameterList Type "::<=" Value

**ParameterizedValueSetTypeAssignment ::=**  
 typereference ParameterList Type "::<=" ValueSet

**ParameterizedObjectClassAssignment ::=**  
 objectclassreference ParameterList "::<=" ObjectClass

**ParameterizedObjectAssignment ::=**  
 objectreference ParameterList DefinedObjectClass "::<=" Object

**ParameterizedObjectSetAssignment ::=**  
 objectsetreference ParameterList DefinedObjectClass "::<=" ObjectSet

**ParameterList ::=** "{" Parameter "," + "}"

**Parameter ::= ParamGovernor ":" DummyReference | DummyReference**  
**ParamGovernor ::= Governor | DummyGovernor**  
**Governor ::= Type | DefinedObjectClass**  
**DummyGovernor ::= DummyReference**  
**DummyReference ::= Reference**  
**ParameterizedReference ::=**  
    **Reference | Reference "{" "}"**  
**SimpleDefinedType ::= Externaltypereference | typereference**  
**SimpleDefinedValue ::= Externalvaluereference | valuereference**  
**ParameterizedType ::= SimpleDefinedType ActualParameterList**  
**ParameterizedValue ::= SimpleDefinedValue ActualParameterList**  
**ParameterizedValueSetType ::= SimpleDefinedType ActualParameterList**  
**ParameterizedObjectClass ::= DefinedObjectClass ActualParameterList**  
**ParameterizedObjectSet ::= DefinedObjectSet ActualParameterList**  
**ParameterizedObject ::= DefinedObject ActualParameterList**  
**ActualParameterList ::= "{" ActualParameter "," + "}"**  
**ActualParameter ::= Type | Value | ValueSet | DefinedObjectClass | Object | ObjectSet**