Source:SA5 (Telecom Management)Title:2 Rel-6 CR 32.601/3 Basic CM IRP Requirements / CORBA SSDocument for:ApprovalAgenda Item:7.5.3

Doc1stevel	Specific a	CR	R	Phase	Subject		VersCu	Doc2ndLev	Workitemsl D
SP-040806	32.601	002		Rel-6	Add Signalling Transport Network (STN) NRM IRP in 32.601 BasicCM IRP Requirements	В	6.0.0	S5-049040	OAM-NIM
SP-040806	32.603	015		Rel-6	Align the IDL style in the CORBA SS with the IDL Style Guide in 32.150	F	6.1.0	S5-047125	OAM-NIM

3GPP TSG-SA5 (Telecom Management) Meeting #40, Sanya, CHINA, 15 - 19 November 2004

S5-049040

	CHANGE REQUEST
æ	32.601 CR 002 # rev - [#] Current version: 6.0.0
For <u>HELP</u> on us	ing this form, see bottom of this page or look at the pop-up text over the $\frac{3}{3}$ symbols.
Proposed change a	ffects: UICC apps # ME Radio Access Network X Core Network X
Title: #	Add Signalling Transport Network (STN) NRM IRP in 32.601 BasicCM IRP Requirements
Source: ೫	SA5 (<u>Ilrui@bupt.edu.cn;</u> liyewen@chinamobile.com)
Work item code: 🕱	OAM-NIM Date: # 19/11/2004
Category: ℜ	B Release: Rel-6 Use one of the following categories: Use one of the following releases: 2 F (correction) 2 (GSM Phase 2) A (corresponds to a correction in an earlier release) R96 (Release 1996) B (addition of feature), R97 (Release 1997) C (functional modification of feature) R98 (Release 1998) D (editorial modification) R99 (Release 1999) Detailed explanations of the above categories can Rel-4 (Release 4) be found in 3GPP TR 21.900. Rel-5 (Release 5) Rel-6 (Release 6) Rel-6 Rel-6
Reason for change:	Since Signalling Transport Network (STN) NRM IRP is introduced in R6, the scope of NRM IRP which can use BasicCMIRP needs to be extended.
Summary of change	BasicCMIRP should be extended to be applicable to new NRM model, such as Signalling Transport Network (STN) NRM IRP.
Consequences if not approved:	光
Clauses affected:	¥ 2, 4.1
Other specs affected:	Y N X Other core specifications X X Test specifications X X O&M Specifications
Other comments:	¥

Change in Clause 2

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] 3GPP TS 32.101: "Telecommunication management; Principles and high level requirements".
- [2] 3GPP TS 32.102: "Telecommunication management; Architecture".
- [3] 3GPP TS 32.600: "Telecommunication management; Configuration Management (CM); Concept and high-level requirements".
- [4] 3GPP TS 32.622: "Telecommunication management; Configuration Management (CM); Generic network resources Integration Reference Point (IRP): Network Resource Model (NRM)".
- [5] 3GPP TS 32.632: "Telecommunication management; Configuration Management (CM); Core Network Resources Integration Reference Point (IRP): Network Resource Model (NRM)".
- [6] 3GPP TS 32.642: "Telecommunication management; Configuration Management (CM); UTRAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".
- [7] 3GPP TS 32.652: "Telecommunication management; Configuration Management (CM); GERAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".
- [8]
 3GPP TS 32.742: "Telecommunication management; Configuration Management (CM);

 Signalling Transport Network (STN) Interface Network Resource Model (NRM) Integration

 Reference Point (IRP): Information Service (IS)".

End of Change in Clause 2

Change in Clause 4.1

4.1 General Requirements

This requirements specification defines requirements for the IS for this IRP. As such, capabilities specified here as being required in the IS are not necessarily required in the product implementation. That which is required in the product implementation will be specified in the IS itself.

The following general and high-level requirements shall apply for the present IRP:

- A. IRP-related requirements in 3GPP TS 32.101 [1].
- B. IRP-related requirements in 3GPP TS 32.102 [2].
- C. IRP-related requirements in 3GPP TS 32.600 [3].

In addition to the above, the following more specific requirements shall apply:

- 1. The IS defined by this IRP shall enable an NM to operate on (access) any of the NRMs defined in [4], [5], [6], and [7] and [8].
- 2. The IS defined by this IRP shall as far as possible be independent of any specific definitions of MOCs, attributes etc. in the NRMs referred to in item 1.
- 3. The IS specified by this IRP shall assume that when this IRP is implemented that the Kernel CM IRP is also implemented.

End of Change in Clause 4.1 End of the Document

3GPP TSG-SA5 (Telecom Management) Meeting #40, Sanya, CHINA, 15 - 19 November 2004

S5-047125

weeting #40, Sai	nya, u	CHIN	IA, 1:) - 19 N	oven	nber .	2004							
			C	HAN	GE	REC	QUE	EST						CR-Form-v7
₩	32.6	6 <mark>03</mark>	CR	015		⊮rev	-	æ	Currer	nt vers	ion:	<mark>6.1</mark>	.0	೫]
For <u>HELP</u> on us	sing th	is forr	n, see	bottom o	of this	page c	or look	at th	е рор-и	ıp text	over	the ₩	syn	nbols.
Proposed change a	affects	s: U	ICC a	pps₩]	ME	Ra	idio A	ccess N	Vetwor	k X	Cor	e Ne	twork X
Title: ೫	Align	the I	DL sty	le in the (CORE	ASS	with th	e IDL	Style (Guide	<mark>in 32</mark> .	150		
Source: ೫	SA5	ZTE	(<mark>huang</mark>	<mark>sq@zte.</mark>	com.c	<mark>n, xior</mark>	<mark>gkj@</mark> :	zte.co	m.cn)					
Work item code: ೫	OAN	<mark>1-NIM</mark>							Da	ate: ೫	19/	<mark>11/20</mark>	04	
Category: ¥	F Use <u>or</u> F A C D Detaile be fou	ne of ta (corra (corra (adda (func (edita ed exp nd in 3	he follo ection) esponc ition of tional r orial mo lanatio 3GPP <u>1</u>	wing cates ls to a confeature), nodification, ns of the a R 21.900,	gories: rection on of fe) above o	<i>in an e</i> ature) categor	earlier i ies car	release	Relea Use 2 e) R R R R R R R	se: # one of 96 97 98 99 el-4 el-5 el-6	Rel (GSN (Rele (Rele (Rele (Rele (Rele (Rele (Rele	<mark>-6</mark> Ilowing ase 19 ase 19 ase 19 ase 19 ase 5 ase 5 ase 6	g rele se 2) 996) 997) 998) 999)))	pases:
Reason for change	:: ¥	Add t Basic	he ma CMIR	ndatory e P. Align	except	ion ope yle wit	eratior h IDL	nNotS Style	Supporte Guide.	ed for	optio	nal op	perati	ions in
Summary of chang	' e: ₩	Add t Basic	he ma CMIRI	ndatory e P. Align	except IDL st	<mark>ion ope</mark> yle wit	eration h IDL	nNotS Style	Supporte Guide.	ed for	optio	nal op	oerati	ions in
Consequences if not approved:	æ	Basic suppo IRPM	CMIR ort the anage	P can not optional er.	t throv opera	/ the st tions.	tandar This r	d-def	ined ex andard	ceptio behav	n whe	en it c can c	loes onfus	not se
0	0.0	0.0.4		٨										
Clauses affected: Other specs affected:	*	6.3, A Y N X X X	Other Test s O&M	A core spe specificat Specifica	ecificat ions ations	tions	¥							
Other comments:	ж													

6.3 Operation parameter mapping

The Basic CM IRP: IS (see 3GPP TS 32.602 [4]) defines semantics of parameters carried in operations across the Basic Configuration Management IRP. Tables 2 through 8 indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

The SS operation find_managed_objects is equivalent to the IS operation getMoAttributes when called with ResultContents set to NAMES_AND_ATTRIBUTES. Iterating the BasicCmInformationIterator is used to fetch the result.

••••

Change in Table 3

Table 3: Mapping from IS getContainment parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
invokeldentifier	- (No equivalence)	-
invokeldentifierOut	Return value of type BasicCmInformationIterator	М
baseObjectInstance	in DN baseObject	М
scope	in SearchControl searchControl (SearchControl.type and SearchControl.level)	0
Not specified in IS	in SearchControl searchControl (SearchControl.filter)	М
containment	Return value of type BasicCmInformationIterator - parameter out ResultSet	М
-4-4		B.4
status	Exceptions:	IVI
	FindmanagedObjects,	
	ManageoGenericiRPSystem::OperationNotSupported,	
	ManageoGenericiRPSystem::ParameterNotSupported,	
	IllegalDNFormatException,	
	UndefinedScopeException,	
	IllegalScopeTypeException,	
	IllegalScopeLevelException,	
	IllegalFilterFormatException,	
	FilterComplexityLimit	

End of change in Table 3

Change in Table 5

Table 5: Mapping from IS cancelOperation parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
invokeldentifier	- (Not applicable, the BasicCmInformationIterator instance identifies the ongoing	М
	operation)	
status	Exceptions:	М
	ManagedGenericIRPSystem::OperationNotSupported,	
	DestroyException	

End of change in Table 5

Change in A

```
Annex A (normative):
CORBA IDL, Access Protocol
#ifndef BASICCMIRPSYSTEM IDL
#define BASICCMIRPSYSTEM IDL
#include "ManagedCenericIRPConstDefs.idl"
#include "ManagedCenericIRPSystem.idl"
// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"
module BasicCmIRPSystem
£
  * Defines the name of a Managed Object Class
   * /
  typedef string MOClass;
 /**
* The format of Distinguished Name (DN) is specified in 3GPP TS 32.300
 * "Name Conventions for Managed Objects".
 */
 -typedef string DN;
/**
* Defines the name of an attribute of a Managed Object
 */
typedef string MOAttributeName;
  /**
   * Defines the value of an attribute of a Managed Object in form of a CORBA
   * Any. Apart from basic datatypes already defined in CORBA, the allowed
 * attribute value types are defined in the AttributeTypes module.
 */
 typedef any MOAttributeValue;
   / * *
   *
      This module adds datatype definitions for types
   * used in the NRM which are not basic datatypes defined
  * already in CORBA.
  */
 - module AttributeTypes
 <del>__</del>{
     / * *
        - A set of strings.
   ____*/
   typedef sequence<string> StringSet;
____;÷
    exception IllegalFilterFormatException {
      string reason;
    ++
```

```
-exception IllegalDNFormatException {
    -string reason;
<del>};</del>
 -exception IllegalScopeTypeException {
   <del>};</del>
  exception IllegalScopeLevelException {
    <del>string reason;</del>
  <del>};</del>
 ____<del>};</del>
______string_reason;
<del>};</del>
 -exception FilterComplexityLimit {
<del>};</del>
 -exception DuplicateMO {};
 -exception CreateNotAllowed {};
 -exception ObjectClassMismatch {};
 -exception NoSuchObjectClass {
 MOClass objectClass;
<u>___</u>};
 -exception ParentObjectDoesNotExist {};
  / * *
   * System otherwise fails to complete the operation. System can provide
  -* reason to qualify the exception. The semantics carried in reason
  * is outside the scope of this IRP.
  */
 - exception NextBasicCmInformations { string reason; };
-- exception NextDeleteErrors { string reason; };
exception DestroyException { string reason; };
exception FindManagedObjects { string reason; };
-- exception ModifyManagedObjects { string reason; };
<u>/**</u>
 * In this version the only allowed filter value is "TRUE" i.e. a filter that
 * matches everything.
 */
— typedef string FilterType;
/**
  -* ResultContents is used to tell how much information to get back
  * from the find_managed_objects operation.
   *
   * NAMES: Used to get only Distinguished Name
```

```
for MOs.
    *
         The name contains both the MO class
          and the names of all superior objects in the naming
    *
          -tree
   * NAMES_AND_ATTRIBUTES: Used to get both NAMES plus
       MO attributes (all or selected).
   * /
   enum ResultContents
 <del>___{</del>
 NAMES,
 <del>};</del>
 /**
 * ScopeType defines the kind of scope to use in a search
 * together with SearchControl.level, in a SearchControl value.
  *
  - * SearchControl.level is always >= 0. If a level is bigger than the
  * depth of the tree there will be no exceptions thrown.
  * BASE_ONLY: level ignored, just return the base object.
  * BASE NTH LEVEL: return all subordinate objects that are on "level"
  - * distance from the base object, where 0 is the base object.
  - * BASE_SUBTREE: return the base object and all of its subordinates
 * down to and including the nth level.
  * BASE_ALL: level ignored, return the base object and all of it's
 * subordinates.
  */
 <u>enum ScopeType</u>
  -{
     BASE_ONLY,
     BASE_NTH_LEVEL,
     BASE_SUBTREE,
     BASE_ALL
  ++
  /**
   -* SearchControl controls the find_managed_object search,
   * and contains:
   * the type of scope ("type" field),
   -* the level of scope ("level" field), level 0 means the "baseObject",
   * level 1 means baseobject including its sub-ordinates etc..
   * the filter ("filter" field),
   * the result type ("contents" field).
  - * The type, level and contents fields are all mandatory.
  * The filter field contains the filter expression.
  * The string "TRUE" indicates "no filter",
  * i.e. a filter that matches everything.
  */
<del>___{</del>
     -ScopeType type;
 unsigned long level;
ResultContents contents;
<del>};</del>
/**
  * Represents an attribute: "name" is the attribute name
   * and "value" is the attribute value.
   * /
  struct MOAttribute
```

```
- MOAttributeName name;
   <u>MOAttributeValue value;</u>
<del>_};</del>
 typedef sequence<MOAttribute> MOAttributeSet;
  struct Result
 -
     DN mo;
   <u>MOAttributeSet attributes;</u>
 <del>};</del>
 typedef sequence<Result> ResultSet;
  / * *
  * AttributeErrorCategory defines the categories of errors, related to
 * attributes, that can occur during creation or modification of MOs.
  *
  * NO SUCH ATTRIBUTE: The specified attribute does not exist.
 * INVALID_ATTRIBUTE_VALUE: The specified attribute value is not valid.
  <u>* MISSINC_ATTRIBUTE_VALUE: An attribute value is required but none was</u>
 *
      provided and no default value is defined for the attribute.
 * INVALID_MODIFY_OPERATOR: The specified modify operator is not valid
 *
      (e.g. operator ADD_VALUES applied to a non multi valued attribute
   * or operator SET_TO_DEFAULT applied where no default value is defined).
   -* MODIFY_NOT_ALLOWED; The modification of the attribute is not allowed.
  * MODIFY FAILED: The modification failed because of an unspecified reason.
  */
   enum AttributeErrorCategory
   ╉
     NO_SUCH_ATTRIBUTE,
     -INVALID_ATTRIBUTE_VALUE,
    - MODIFY_FAILED
  <del>};</del>
 / * *
   * DeleteErrorCategory defines the categories of errors that can occur
   * during deletion of MOs.
   *
 * SUBORDINATE OBJECT: The MO cannot be deleted due to subordinate MOS.
  * DELETE NOT ALLOWED: The deletion of the MO is not allowed.
 * DELETE FAILED: The deletion failed because of an unspecified reason.
 */
 - enum DeleteErrorCategory
---{
     SUBORDINATE OBJECT,
 DELETE_NOT_ALLOWED,
  _____DELETE_FAILED
<del>};</del>
 /**
     AttributeError represents an error, related to an attribute, that occured
   * during creation or modification of MOs.
   <u>* It contains:</u>
     the name of the indicted attribute ("name" field),
```

* - the category of the error ("error" field),
* - optionally the indicted attribute value ("value" field)
* - optionally, additional details on the error ("reason" field)
+/
struct AttributeError
MOAttributeName name;
AttributeErrorCategory_error:
- Monteribucevarue varuer
string reason;
<pre>typedef sequence<attributeerror> AttributeErrorSeq;</attributeerror></pre>
* DeleteError represents an error that occured during deletion of MOs.
* It contains:
* - the distinguished name of the indicted MO ("objectName" field),
* entionally additional details on the error ("reagon" field)
- optionarry, additional details on the error (reason freid).
*/
<pre>typedef sequence<deleteerror> DeleteErrorSeq;</deleteerror></pre>
/**
ModifyAttributeErrors represents errors that occured during
* modification of attributes of a MO.
* It contains:
* - the distinguished name of the indicted MO ("objectName" field),
* - a sequence containing the attribute errors ("errors" field)
*/
- Struct ModifyAttributeEffors
DN objectName;
AttributeErrorSeq errors;
_
tymodof, gomenna Modify Ittributo Emerges, Modify Ittributo Emerges (Com
<u> /**</u>
/** The BasicCmInformationIterator is used to iterate through a snapshot of
/** ———————————————————————————————————
— /** — The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager used it to page the network of Managed Object Information.
—/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information.
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information.</pre>
<pre>/**</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop</pre>
/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration.
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator { } </pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator { } </pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator (**</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator { /** This method returns between 1 and "how many" Managed Object information</pre>
<pre>/** The BasicCmInformationIterator is used to iterate through a snapshot of Managed Object Information when IRPManager invokes find_managed_objects. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface BasicCmInformationIterator [/**</pre>

more items to return. "how many" must be non-zero. Return TRUE if there
may be more Managed Object information to return. Return FALSE if there
are no more Managed Object information to be returned.
If FALSE is returned the IRPAgent will automatically destroy the
itorator
@narm how many how many alements to return in the "fatshadElements" out
- sparin now many crements to return in the retencationents ou
"fetchedElements" is empty when the BasicCmInformationIterator is
<u>*/</u>
<u> </u>
in ungigned short how many
and Degult Sot fot about interpreta
Vale Reputebet Tetenedifements
/**
<u>*/</u>
usid doctrocy ()
- Void destroy ()
<u>/**</u>
/**
— /** — The DeleteResultIterator is used to iterate through the list of deleted MOG — when IRPManager invokes method "delete_managed_objects". IRPManager uses it to page the return of Managed Objects.
—/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information.
/**/**The DeleteResultIterator is used to iterate through the list of deleted MOGwhen IRPManager invokes method "delete_managed_objects"IRPManager uses it to pace the return of Managed Object InformationIRPManager uses it to pace the return of Managed Object Information.
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration.</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator </pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOV when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { </pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** </pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Mod when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** </pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Mod when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MON when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that:</pre>
<pre>/**</pre>
<pre>/**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains only the DNs of the deleted MOs (no attributes are returned). If FALSE is returned, the IRPAgent will not automatically destroy the</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasisCmInformationIterator { /** Inherited method "next_basisCmInformations" has the same behaviour as for interface BasisCmInformationIterator, except that: The Managed Object information returned in parameter</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MON when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface DasicCmInformationIterator, except that:</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOW when IRDManager invokes method "delete_managed_objects". IRDManager uses it to pace the return of Managed Object Information. IRDAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRDManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted More when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator {</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOG when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains only the DNs of the deleted MOs (no attributes are returned). If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** /** /** /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted More when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains only the DNs of the deleted MOs (no attributes are returned). If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" deletion errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Mod when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /**</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Mov when IRPManager invokes method "delete_managed_objects". IRPManager uses it to page the return of Managed Object Information. IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Mod when IRPManager invokes method "delete_managed_objects". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator {</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted MOA when IRPManager invokes method "delete_managed_object". IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformationIterator, except that:</pre>
<pre>/** The DeleteResultIterator is used to iterate through the list of deleted Move when IRPManager invokes method "delete_managed_object Information. IRPManager uses it to pace the return of Managed Object Information. IRPAgent controls the life cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration. */ interface DeleteResultIterator : BasicCmInformationIterator { /** Inherited method "next_basicCmInformations" has the same behaviour as for interface BasicCmInformation returned in parameter "fetchedBlements" contains only the DNs of the deleted MOs (no attributes are returned). If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" deletion errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more deletion errors to be returned. </pre>

destroy the iterator.
<pre>@parm how_many: how many deletion errors to return in the</pre>
@narm fotgbodDolotoFrrong: the dolotion errorg
Oversigned to be been indigeting if one deletion emerging and returned
<pre></pre>
<u> </u>
/**
IRPManager uses it to pace the return of Managed Object Information.
interface MedifyPegultIterator · Pagig@mInformationIterator
/**
Inherited method "next basicCmInformations" has the same behaviour as
for interface BasicCmInformationIterator, except that:
<pre>for interface BasicCmInformationIterator, except that:</pre>
for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs.
<pre>for interface BasicCmInformationIterator, except that:</pre>
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed</pre>
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically</pre>
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. /** //** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator.</pre>
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. /** //** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator. @parm how_many: how many modification errors to return in the "fotabadMedifuErrors" out parameter </pre>
 for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the
 for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non zero. Return TRUE if there are more modification errors to returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator. @parm how_many: how many modification errors. @parm how_many: how many modification errors. @parm fetchedModifyErrors: the modification errors.
<pre> for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator. @parm how_many: how many modification errors. @parm fetchedModifyErrors: the modification errors. @parm fetchedModifyErrors: the modification errors are returned. */</pre>
 for interface BasicCm⁻InformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. *// /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are no more more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator. @parm how_many: how many modification errors to return in the "fetchedModifyErrors: the modification errors are returned. #parm fetchedModifyErrors: the modification errors are returned. #parm fetchedModifyErrors (boolean indicating if any modification errors are returned. */ boolean next_modificationErrors (boolean next_modif
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedElements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-sero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator. @parm how_many: how many modification errors. @parm how_many</pre>
<pre>for interface BasicCmInformationIterator, except that: The Managed Object information returned in parameter "fetchedBlements" contains DNs and attributes of the modified MOs. If FALSE is returned, the IRPAgent will not automatically destroy the iterator. */ /** This method returns between 0 and "how_many" modification errors. The IRPAgent may return less than "how_many" items even if there are more items to return. "how_many" must be non-zero. Return TRUE if there are more modification errors to return. Return FALSE if there are no more modification errors to be returned. If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information. @parm how_many: how many modification errors. @parm how_many: how many modification errors. @parm fetchedModifyErrors: the modification errors are returned. */ boolean next_modificationErrors (</pre>

ManagedGenericIRPSystem::InvalidParameter); }; // end of ModifyResultIterator typedef sequence<MOAttributeName> AttributeNameSet; /** * ModifyOperator defines the way in which an attribute value is to be * applied to an attribute in a modification of MO attributes. * * REPLACE: replace the current value with the provided value * ADD_VALUES: for a multi-valued attribute, add the provided values to the <u>* current list of values</u> * REMOVE_VALUES: for a multi-valued attribute, remove the provided values * from the current list of values -* SET_TO_DEFAULT: set the attribute to its default value */ ___{ REPLACE, ADD_VALUES, REMOVE_VALUES, }; /** * AttributeModification defines an attribute value and the way it is to * be applied to an attribute in a modification of MO attributes. * It contains: the name of the attribute to modify ("name" field), the value to apply to this attribute ("value" field), the way the attribute value is to be applied to the attribute <u>* ("operator" field).</u> * / -struct AttributeModification <u>MOAttributeName name;</u> <u>MOAttributeValue value;</u> ModifyOperator operator; }; /** * The BasicCmIrpOperations interface. * Supports a number of Resource Model versions. */ ----{ /** * Get the version(s) of the interface * * @raises GetBasicCmIRPVersion when the system for some reason * can not return the supported versions. * @returns all supported versions. * /

/**	
*	Return the operation profile for a specific Basic CM IRP version.
*	
*	@raiges CetRagicCmIRPOperationProfile when the system for some reason
*	- connot return the supported operations and parameters.
*	ereturns the list of all supported operations and their supported
*	
*/	
Man	agedCenericIRPConstDefs::MethodList_get_basicCm_IRP_operation_profile
(
· · · · · · · · · · · · · · · · · · ·	in ManagedGenericIRPConstDefs::VersionNumber basicCm IRP version
<u> </u>	
	ses (GetBasicCmIRPOperationProfile,
	ManagedGenericIRPSystem::OperationNotSupported.
	ManagedGenericIRPSystem::InvalidParameter);
/**	
*	Return the notification profile for a specific Basic CM IRP version.
*	
*	@raises_CetBasicCmIRPNotificationProfile_when_the_avatem_for_some
*	- reagon cannot return the supported potifications and parameters
*	ereturns the list of all supported notifications and their supported
*	- parameters for the specified version
* /	parameters for the specified version.
Man	agedConorigIRDCongtDefg::MethodLigt
	get bagicem IRP notification profile (
	in ManagodConorigIPDCongtDofg. WorgionNumber bagigCm IPD worgion
<u>}</u>	
	and (CotRagiaCmIRPNotificationProfile)
	ManagedGenericIPDSystem::OperationNotSupported
	ManagedConorigIRDSystem::InvalidDarameter):
/**	
<u>*</u>	Derforma a containment coarch using a SearchControl to
*	control the george and the returned regulta
*	concror the search and the retained resurts.
*	All MOg in the grape constitute a get that the filter works on
*	The regult Regig(mInformationIterator contains all matched MOg
*	with the amount of detail apogified in the SoarabControl
*	For the gradial gage when no managed objects are matched in
*	find managed objects the RadiaCmInformationIterator will be returned
*	Executing the post bagig@mInformations in the
*	PagiaCmInformationIterator will return ENLSE for
*	applotion_
*	
*	Proxy bagoObject The start MO in the containment tree
*	@parm goarghControl the SearghControl to use
*	eparm requestedAttributes defines which attributes to get
*	Tf this parameter is empty ("") all attributes deal
*	be returned. In this worsion this is the only supported competies
*	Note that this engument is only
*	rolowant if RegultContents in the search control is
*	apodifed to NAMES AND ATTERPIETS
*	SPECIFICA CO MANDALANA ATTAIDOIDO.
*	
*	Project ManagodConorigIDDCystom . WalyoNotConnected if a walid but
*	eraises managedoenerrerrrystemvaruenotsupported if a varia DUE
	tiold in the georghaphtrol percentation contains the solute NAMES and
X	Hield in the search control parameter contains the value NAMES and
<u>۲</u>	the optional getcontainment is operation is not supported.
т х	WEALBED UNDELLIGEAMOUNCEPTION INC MU GOES NOT EXIST.
<u>×</u>	WEALSCH ILLEGALDNFORMALEXCEPTION THE AN SYNTAX SERING 15
X	mallormed.

* @raises IllegalScopeTypeException The ScopeType in scope contains	
* an illegal value	
* main and a local and magnetics The game lovel is possive	
<u>* (<0)</u> .	
* @raises IllegalFilterFormatException The filter string is	
* malformed.	
* @raiges FilterComplexityLimit if the filter syntax is correct.	
* but the filter is the applant to be presented by the TBD agent	
- Dut the lifter is too complex to be processed by the two agent.	
* @see SearchControl	
* @see BasicCmInformationIterator	
<u>*/</u>	
raises (FindManagedObjects,	
ManagedGenericIRPSystem::ParameterNotSupported,	
ManagedGenericIRPSystem:: InvalidParameter.	
ManagodConorigIDBCystom · ValueNotSupported	
UndefinedmotixCeption,	
UndefinedScopeException,	
The second	
FilterComplexityLimit);	
/ <u>* *</u>	
* Porforms the greation of a MO instance in the MIR maintained	
* he the TDR end	
<u>*</u>	
* @parm objectName: the distinguished name of the MO to create.	
* @parm referenceObject: the distinguished name of a reference MO.	
* enarm attributor. in input initial attribute values for the MO to	
* eparm attributes: in input, initial attribute values for the MO to	
<pre>* @parm attributes: in input, initial attribute values for the MO to create; in output, actual attribute values of the created MO.</pre>	
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	Ð
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	Ð
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	Ð
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	d) d
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	d
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	Ð Ð
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	đ
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	e e
<pre></pre>	e
<pre></pre>	d
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	2
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	d
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	c
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedGenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one</pre>	2
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create.</pre>	c
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	d) di
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises CreateNotAllowed: The creation of the MO is not allowed. </pre>	d
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedGenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create doe * eraises ObjectClassMismatch: The object of the more is for the spin of the spin of the create of the spin of the create of the cr</pre>	6
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	2 €
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	2 2
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedGenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create doe * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized.</pre>	2 f:
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	2
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises ObjectClassMismatch: The object class of the MO to create doe * match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * @raises ParentObjectDoceNotExist: The parent MO instance of the * @raises ParentObjectDoceNotExist: The parent MO instance of the * @raises ParentObjectDoceNotExist: The parent MO instance of the * @raises ParentObjectDoceNotExist: The parent MO instance of the * @raises ParentObjectDoceNotExist: The parent MO instance of the * @raises ParentObjectDoceNotExist: The parent MO instance of the * # ManagedExist # MonagedExist # MO for an exist. * @raises ParentObjectDoceNotExist: The parent MO instance of the * # ManagedExist # ManagedExist # MonagedExist # MonagedExist# # Monag</pre>	2 2
<pre>* @parm attributes: in input, initial attribute values for the MO to * @reate; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The OD does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises ObjectClassHismatch: The object class of the MO to create doe * match with the object class of the object to create is not * @raises ParentObjectDeesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. * * recognized. * @raises ParentObjectDeesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. </pre>	2 a
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises IllegalDNFormatException: The DN syntax string is malformed. * deraises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassWismatch: The object class of the MO to create doc * mot match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoceNotExist: The parent MO instance of the * managedEntity specified to be created doce not exist. */</pre>	2 ft
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * eraises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises ObjectClassHismatch: The object class of the MO to create doc * not match with the object class of the opiest to create is not * recognized. * @raises ParentObjectDeeoNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. * @raises ParentObjectDeeoNotExist: The parent MO instance of the * Managed_object (</pre>	2 ft
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedCenerieIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenerieIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenerieIRPSystem::InvalidParameter: An invalid * parameter is not supported. * @raises UndefinedMOException: The MO does not exist. * @raises IllegalDNFormatException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises ObjectClassMismatch: The object class of the MO to create doe * not match with the object class of the opicet to create is not * @raises ParentObjectClass: The class of the object to create is not * # @raises ParentObjectClass: The class of the object to create is not * # @raises ParentObjectClass: The class of the object to create is not * # @raises ParentObjectClass: The class of the object to create is not * # @raises ParentObjectClass: The class of the object to create is not * # @raises ParentObjectDecoNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>	2 A
<pre>* @parm attributes: in input, initial attribute values for the M0 to * @parm attributeErrors: errors, related to attributes, that caused the * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * @raises ManagedCenericIRPSystem::ParameterNotSupported: An optional * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid * @raises UndefinedMOException: The M0 does not exist. * @raises UndefinedMOException: The M0 does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises DuplicateMO: A M0 already exist with the same DN as the one * to create. * @raises ObjectClassMismatch: The object class of the M0 to create doe * @raises NoSuchObjectClass: The class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent M0 instance of the * ManagedEntity specified to be created does not exist. * #/ void create_managed_object (</pre>	2 A
<pre>* @parm attributes: in input, initial attribute values for the MO to * ereate; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * ereation of the MO to fail. * * @raises ManagedCenericIRPSystem::OperationNotSupported: The operation *is not supported. * @raises ManagedCenericIRPSystem::DarameterNotSupported: An optional *parameter is not supported. * @raises ManagedCenericIRPSystem::InvalidParameter: An invalid *parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises ObjectClassMismatch: The object class of the MO to create doe * not match with the object class of the provided reference MO. * @raises NeuchObjectClass: The class of the provided reference MO. * @raises ParentObjectDoesNotExist: The parent MO instance of the * recognized. * ManagedEntity specified to be created does not exist. */ woid create_managed_object (</pre>	2
<pre>* @parm attributes: in input, initial attribute values for the NO to * create; in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * erraises ManagedCenerieIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedCenerieIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenerieIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises CreateNotAllowed: The object class of the MO to create doe</pre>	2
<pre>* @parm attributes: in input, initial attribute values for the MO to * create; in output, actual attribute values of the ereated MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * @raises ManagedGenerieIRPSystem::OperationNotSupported: The operation * is not supported. * @raises ManagedGenerieIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenerieIRPSystem::ParameterNotSupported: An optional * parameter is not supported. * @raises ManagedCenerieIRPSystem::InvalidParameter: An invalid * parameter value has been provided. * @raises UndefinedMOException: The MO does not exist. * @raises UndefinedMOException: The DN syntax string is malformed. * @raises DuplicateMO: A MO already exist with the same DN as the one * to create. * @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create doe * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recegnized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>	2 2
<pre>* @parm attributes: in input, initial attribute values for the MO to * create: in output, actual attribute values of the created MO. * @parm attributeErrors: errors, related to attributes, that caused the * creation of the MO to fail. * * eraises ManagedCenerieIRPEystem::OperationNotSupported: The operation * is not supported. * eraises ManagedCenerieIRPEystem::ParameterNotSupported: An optional * parameter is not supported. * eraises ManagedCenerieIRPEystem::InvalidParameter: An invalid * parameter value has been provided. * eraises UndefinedMOException: The MO does not exist. * eraises UndefinedMOException: The DN syntax string is malformed. * eraises DuplicateMO: A MO already exist with the same DN as the one * eraises ObjectClassMismatch: The object class of the MO to create doe * oreate. * eraises ObjectClassMismatch: The object class of the MO to create doe * and match with the object class of the experiment of the * eraises ParentObjectClass: The class of the object to create is not * creases ParentObjectClass: The parent MO instance of the * managedEntity specified to be oreated does not exist. * woid create_managed_object. * ManagedEntity specified to be oreated does not exist. */ * void create_managed_object. * in DN referenceObject, in DN referenceObject, out AttributeErrorSeq attributeErrors</pre>	2 A
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	2 2
<pre>* @parm attributes: in input, initial attribute values for the MO to</pre>	e e

ManagedGenericIRPSystem::ParameterNotSupported,
ManagedGenericIRPSystem::InvalidParameter,
UndefinedMOException,
CreateNotAllowed,
/**
* Performs the deletion of one or more MO instances from the MIB
* maintained by the IRPAgent, using a SearchControl to control the
* instances to be deleted.
· · · · · · · · · · · · · · · · · · ·
* All MOs in the scope constitute a set that the filter works on.
All matched MOs will be deleted by this operation.
* The returned DeleteResultIterator is used to retrieve the DNs of the
* MOs deleted and the errors that may have occurred preventing deletion
* Executing the next_basicCmInformations in the DeleteResultIterator
* will return FALSE for completion.
· · · · · · · · · · · · · · · · · · ·
* @parm baseObject: the start MO in the containment tree.
* @parm searchControl: the SearchControl to use; field "contents" has no
* meaning here and shall be ignored.
· · · · · · · · · · · · · · · · · · ·
* @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
* 15 not supported.
parameter value has been provided. t out the Wolfstein dwolfstein the Wolfstein the Wolfst
<pre> @raises IllegalDNFormatException: The DN syntax string is mailormed. * Overing IllegalComeException: The ComeExact in second sectors </pre>
* en illegal value
* empired Illegal GameLevelEvention: The game level is negative (20)
* @raises IllegalScopeLevelException. The scope level is negative ().</td
* eraises Filter Complexity init: The filter arter is correct
* but the filter is the someley to be presented by the IDDAgent
*/
DeleteRegultIterator delete managed objects (
in DN hagoObject
in SoorabControl goorabControl
ManagedConcrigIRPSystem::OperationNotSupported
ManagedGenericIRPSystem: OperationNotSupported,
FilterComplexityLimit);
/**
* Performs the modification of MO attributes. One or more MOs attributes
* may be modified according to a SearchControl.
<u>*</u>
* All MOs in the scope constitute a set that the filter works on.

* All matched MOG will have their attributed modified by this operation
* The returned ModifyPergultIterator is used to retrieve the DNg of the
* modified Most barather with the values of the modified attributes and
* modified most together with the values of the modified attributes, and
the errors that may have occurred preventing modification of some
*-attributes.
* For the special case when no managed objects are matched in
* modify_managed_objects, the ModifyResultIterator will be returned.
* Executing the next_basicCmInformations in the ModifyResultIterator
* will return FALSE for completion.
<u>*</u>
* @parm baseObject: the start MO in the containment tree.
* @parm searchControl: the SearchControl to use; field "contents" has no
meaning here and shall be ignored.
* @parm modifications: the values for the attributes to modify and
the way those values are to be applied to the attributes.
<u>*</u>
* @raigog ManagodConorigIPDSystom: OperationNetSupported: The operation
* ig not gupported
* Sweiges MenagedConcrigTDDGuster: TruslidDeremeter: An invelid
- Wraises Managedener Frikksystem - Invariarameter - An Invaria
* @raises IllegalScopeTypeException: The ScopeType in scope contains
* @raises IllegalFilterFormatException: The filter string is malformed.
* @raises FilterComplexityLimit: The filter syntax is correct,
* but the filter is too complex to be processed by the IRPAgent.
*/·
in DN baseObject,
in SearchControl searchControl,
ManagedGenericIRPSystem::OperationNotSupported.
ManagedGenericIRPSystem::InvalidParameter.
UndefinedMOException.
IlegalDNFormatException_
Indefined&geneExgention_
The subsect of the subsection
Filtercompiexitybimit/r
1.
$\frac{1}{1}$
#endl1

Annex A (normative): CORBA IDL, Access Protocol

A.1 IDL specification (file name "BasicCMIRPConstDefs.idl")

// File: BasicCMIRPConstDefs.idl

#ifndef _BASICCMIRPCONSTDEFS_IDL_ #define _BASICCMIRPCONSTDEFS_IDL_ // This statement must appear after all include statements #pragma prefix "3gppsa5.org" /* ## Module: BasicCMIRPConstDefs This module contains commonly used definitions for BasicCMIRP. _____ */ module BasicCMIRPConstDefs { /** * Defines the name of a Managed Object Class */ typedef string MOClass; /** * The format of Distinguished Name (DN) is specified in 3GPP TS 32.300 * "Name Conventions for Managed Objects". */ typedef string DN; /** * Defines the name of an attribute of a Managed Object * / typedef string MOAttributeName; / * * * Defines the value of an attribute of a Managed Object in form of a CORBA * Any. Apart from basic datatypes already defined in CORBA, the allowed * attribute value types are defined in the AttributeTypes module. */ typedef any MOAttributeValue; /** * $\overline{}^*$ In this version the only allowed filter value is "TRUE" i.e. a filter that * matches everything. */ typedef string FilterType; * ResultContents is used to tell how much information to get back * from the find_managed_objects operation. * * NAMES: Used to get only Distinguished Name for MOs. * * The name contains both the MO class * and the names of all superior objects in the naming * tree. * * NAMES_AND_ATTRIBUTES: Used to get both NAMES plus * MO attributes (all or selected). */ enum ResultContents { NAMES, NAMES_AND_ATTRIBUTES };

```
/**
  * ScopeType defines the kind of scope to use in a search
  * together with SearchControl.level, in a SearchControl value.
  *
  * SearchControl.level is always >= 0. If a level is bigger than the
  * depth of the tree there will be no exceptions thrown.
  * BASE_ONLY: level ignored, just return the base object.
  * BASE_NTH_LEVEL: return all subordinate objects that are on "level"
  * distance from the base object, where 0 is the base object.
  * BASE_SUBTREE: return the base object and all of its subordinates
  * down to and including the nth level.
  * BASE_ALL: level ignored, return the base object and all of it's
  * subordinates.
 */
 enum ScopeType
{
   BASE_ONLY,
    BASE_NTH_LEVEL,
    BASE SUBTREE,
    BASE ALL
 };
 /**
  * SearchControl controls the find_managed_object search,
  * and contains:
  * the type of scope ("type" field),
  * the level of scope ("level" field), level 0 means the "baseObject",
  *
    level 1 means baseobject including its sub-ordinates etc..
  * the filter ("filter" field),
  * the result type ("contents" field).
  * The type, level and contents fields are all mandatory.
  * The filter field contains the filter expression.
  *
   The string "TRUE" indicates "no filter",
  * i.e. a filter that matches everything.
  */
 struct SearchControl
 {
    ScopeType type;
   unsigned long level;
   FilterType filter;
   ResultContents <u>contents;</u>
 };
 /**
  * Represents an attribute: "name" is the attribute name
  * and "value" is the attribute value.
 */
struct MOAttribute
   MOAttributeName name;
  MOAttributeValue value;
};
 typedef sequence <MOAttribute> MOAttributeSet;
 struct Result
    DN mo;
    MOAttributeSet attributes;
 };
 typedef sequence <Result> ResultSet;
```

```
/**
 * AttributeErrorCategory defines the categories of errors, related to
 * attributes, that can occur during creation or modification of MOs.
 *
 * NO_SUCH_ATTRIBUTE: The specified attribute does not exist.
 * INVALID_ATTRIBUTE_VALUE: The specified attribute value is not valid.
 * MISSING_ATTRIBUTE_VALUE: An attribute value is required but none was
   provided and no default value is defined for the attribute.
 * INVALID_MODIFY_OPERATOR: The specified modify operator is not valid
 * (e.g. operator ADD_VALUES applied to a non multi-valued attribute
* or operator SET_TO_DEFAULT applied where no default value is defined).
 * MODIFY_NOT_ALLOWED: The modification of the attribute is not allowed.
 * MODIFY_FAILED: The modification failed because of an unspecified reason.
 */
enum AttributeErrorCategory
{
   NO_SUCH_ATTRIBUTE,
   INVALID ATTRIBUTE_VALUE,
   MISSING ATTRIBUTE VALUE,
   INVALID MODIFY OPERATOR,
   MODIFY NOT ALLOWED,
   MODIFY_FAILED
};
 * DeleteErrorCategory defines the categories of errors that can occur
 * during deletion of MOs.
 *
 * SUBORDINATE_OBJECT: The MO cannot be deleted due to subordinate MOs.
 * DELETE_NOT_ALLOWED: The deletion of the MO is not allowed.
 * DELETE_FAILED: The deletion failed because of an unspecified reason.
 */
enum DeleteErrorCategory
{
   SUBORDINATE_OBJECT,
   DELETE_NOT_ALLOWED,
  DELETE_FAILED
};
/**
 * AttributeError represents an error, related to an attribute, that occured
 * during creation or modification of MOs.
 * It contains:
 * - the name of the indicted attribute ("name" field),
 * - the category of the error ("error" field),
 * - optionally, the indicted attribute value ("value" field),
 * - optionally, additional details on the error ("reason" field).
 */
struct AttributeError
{
   MOAttributeName name;
   AttributeErrorCategory error;
   MOAttributeValue value;
   string reason;
};
typedef sequence <AttributeError> AttributeErrorSeq;
/**
 * DeleteError represents an error that occured during deletion of MOs.
 * It contains:
 * - the distinguished name of the indicted MO ("objectName" field),
```

* - the category of the error ("error" field), * - optionally, additional details on the error ("reason" field). */ struct DeleteError { DN objectName; DeleteErrorCategory error; string reason; }; typedef sequence <DeleteError> DeleteErrorSeq; / * * * ModifyAttributeErrors represents errors that occured during * modification of attributes of a MO. * It contains: - the distinguished name of the indicted MO ("objectName" field), * - a sequence containing the attribute errors ("errors" field). */ struct ModifyAttributeErrors { DN objectName; AttributeErrorSeg errors; }; typedef sequence <ModifyAttributeErrors> ModifyAttributeErrorsSeq; typedef sequence <MOAttributeName> AttributeNameSet; * ModifyOperator defines the way in which an attribute value is to be * applied to an attribute in a modification of MO attributes. * * REPLACE: replace the current value with the provided value * ADD_VALUES: for a multi-valued attribute, add the provided values to the * current list of values * REMOVE_VALUES: for a multi-valued attribute, remove the provided values * from the current list of values * SET_TO_DEFAULT: set the attribute to its default value */ enum ModifyOperator { REPLACE, ADD_VALUES, REMOVE_VALUES, SET TO DEFAULT }; /** * AttributeModification defines an attribute value and the way it is to * be applied to an attribute in a modification of MO attributes. * It contains: * - the name of the attribute to modify ("name" field), * - the value to apply to this attribute ("value" field), * - the way the attribute value is to be applied to the attribute * ("operator" field). */ struct AttributeModification MOAttributeName name; MOAttributeValue value; ModifyOperator operator; };

typedef sequence <AttributeModification> AttributeModificationSet;

};
#endif _BASICCMIRPCONSTDEFS_IDL_

A.2 IDL specification (file name "BasicCMIRPSystem.idl")

// File: BasicCMIRPSystem.idl

#ifndef _BASICCMIRPSYSTEM_IDL_
#define _BASICCMIRPSYSTEM_IDL_

#include "ManagedGenericIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"
#include "BasicCMIRPConstDefs.idl"

// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"

module BasicCmIRPSystem

{

```
exception IllegalFilterFormatException {
    string reason;
 };
 exception IllegalDNFormatException {
   string reason;
 };
 exception IllegalScopeTypeException {
    string reason;
 };
 exception IllegalScopeLevelException {
   string reason;
 };
 exception UndefinedMOException {
    string reason;
 };
exception UndefinedScopeException {
    string reason;
};
exception FilterComplexityLimit {
   string reason;
};
exception DuplicateMO {};
exception CreateNotAllowed {};
exception ObjectClassMismatch {};
exception NoSuchObjectClass {
   BasicCMIRPConstDefs::MOClass objectClass;
};
 exception ParentObjectDoesNotExist {};
/**
```

* System otherwise fails to complete the operation. System can provide
* reason to qualify the exception. The semantics carried in reason
* is outside the scope of this IRP.
*/
<pre>exception NextBasicCmInformations { string reason; };</pre>
<pre>exception NextDeleteErrors { string reason; };</pre>
<pre>exception NextModifyErrors { string reason; };</pre>
<pre>exception DestroyException { string reason; };</pre>
<pre>exception GetBasicCmIRPVersion { string reason; };</pre>
<pre>exception GetBasicCmIRPOperationProfile { string reason; };</pre>
<pre>exception GetBasicCmIRPNotificationProfile { string reason; };</pre>
<pre>exception FindManagedObjects { string reason; };</pre>
<pre>exception CreateManagedObject { string reason; };</pre>
<pre>exception DeleteManagedObjects { string reason; };</pre>
exception ModifyManagedObjects { string reason; };
/**
The BasicCmInformationIterator is used to iterate through a snapshot of
Managed Object Information when IRPManager invokes find managed objects.
IRPManager uses it to pace the return of Managed Object Information.
IRPAgent controls the life-cycle of the iterator. However, a destroy
operation is provided to handle the case where IRPManager wants to stop
the iteration procedure before reaching the last iteration.
*/
interface BasicCmInformationIterator
{
/**
This method returns between 1 and "how many" Managed Object information.
The IRPAgent may return less than "how many" items even if there are
more items to return. "how many" must be non-zero. Return TRUE if there
may be more Managed Object information to return Return FALSE if there
are no more Managed Object information to be returned
If FALSE is returned, the IRPAgent will automatically destroy the
iterator.
@parm how many how many elements to return in the "fetchedElements" out
parameter
@returns A boolean indicating if any elements are returned
"fetchedElements" is empty when the BasicCmInformationIterator is
<u>* /</u>
booloon next bogig@mInformationg (
in ungigned short how many
NextBasiceminiormations,
managedGenericikrSystemOperationNotSupported);
/ * *
<u>inis method destroys the iterator.</u>
void destroy ()

}; // end of BasicCmInformationIterator

/**
The DeleteResultIterator is used to iterate through the list of deleted MOs
when IRPManager invokes method "delete_managed_objects".
IRPManager uses it to pace the return of Managed Object Information.
IRPAgent controls the life-cycle of the iterator. However, a destroy operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration.
{
/**
Inherited method "next_basicCmInformations" has the same behaviour as
for interface BasicCmInformationIterator, except that:
- The Managed Object information returned in parameter
"fetchedElements" contains only the DNs of the deleted MOs
(no attributes are returned).
- If FALSE is returned, the IRPAgent will not automatically destroy the
iterator.
*/
This method returns between U and "now_many" deletion errors. The
IRPAgent may return less than "now_many" items even if there are more
ILENS CO FECULII. "HOW_MAILY" MUSE DE HOH-ZEFO. RECUIII IROE II CHEFE AFE
deletion errors to be returned
defetion errors to be returned.
If FALSE is returned and last call to inherited method "next_basicCmInformations" also returned FALSE (i.e. no more Managed Object information to be returned), the IRPAgent will automatically destroy the iterator.
oner her many deletion errors to return in the
@returns: a boolean indicating if any deletion errors are returned
*/
<u>`</u>
boolean next deleteErrors (
in unsigned short how_many,
out BasicCMIRPConstDefs::DeleteErrorSeq fetchedDeleteErrors
)
raises (
NextDeleteErrors,
ManagedGenericIRPSystem::InvalidParameter);
<pre>}; // end of DeleteResultIterator</pre>
<u>/**</u>
The ModifyResultIterator is used to iterate through the list of modified
MOs when IRPManager invokes method "modify_managed_objects".
IRPManager uses it to pace the return of Managed Object Information.
TRRE sector is the life scale of the interval of the

operation is provided to handle the case where IRPManager wants to stop the iteration procedure before reaching the last iteration.

*/
interface ModifyResultIterator : BasicCmInformationIterator
/**
innerited method "next_basiccminformations" has the same behaviour as
for interface BasicCminformationIterator, except that:
- The Managed Object information returned in parameter
"fetchedElements" contains DNs and attributes of the modified MOs.
- If FALSE is returned, the IRPAgent will not automatically destroy the
iterator.
*/
/ * *
This method returns between 0 and "hew many" medification errors. The
The method feturing between 0 and now_many modification errors. The
ikPAgent may return less than "now_many" items even if there are more
items to return. "how_many" must be non-zero. Return TRUE if there are
more modification errors to return. Return FALSE if there are no more
modification errors to be returned.
If FALSE is returned and last call to inherited method
"next basicCmInformations" also returned FALSE (i.e. no more Managed
Object information to be returned) the IPD/gent will automatically
doctrow the iterator
destroy the iterator.
@parm how_many: how many modification errors to return in the
fetchedModifyErrors" out parameter
@parm fetchedModifyErrors: the modification errors.
@returns: a boolean indicating if any modification errors are returned.
*/
boolean next modificationErrors (
in unsigned short how many
fot abodModifiermone
raises (
NextModifyErrors,
ManagedGenericIRPSystem::InvalidParameter);
}; // end of ModifyResultIterator
/**
* The Register Irron erations interface
- The basic outpoperations incertage.
- Supports a number of Resource Model Versions.
*/
interface_BasicCmIrpOperations
{
/**
* Get the version(s) of the interface
*
* @raises GetBasicCmIPDVersion when the system for some reason
* can not return the supported versions
can not recurn the supported versions.
" @returns all supported versions.
<u> </u>
<pre>ManagedGenericIRPConstDefs::VersionNumberSet get_basicCm_IRP_version()</pre>
raises (GetBasicCmIRPVersion);
/**_
* Return the operation profile for a specific Basic CM IRP version.
*
* @raises GetBasicCmIRPOperationProfile when the system for some reason

*	cannot return the supported operations and parameters.
* @r	eturns the list of all supported operations and their supported
*	parameters for the specified version.
<u>*/</u>	addenericIDDdengtDefa::Methodiist get begigdm IDD eneration profile
Manag	edGenericiRPConstDels. MethodList get_basiccm_iRP_operation_profile
`i	n ManagedGenericIRPConstDefs::VersionNumber basicCm IRP version
)	
raise	s (
G	etBasicCmIRPOperationProfile,
M	anagedGenericIRPSystem::OperationNotSupported,
M	anagedGenericIRPSystem::InvalidParameter);
۲ ۲ (
/**	turn the notification profile for a gradific Degic (M IDD version
^ <u>*</u> Re	turn the notification profile for a specific Basic CM IRP Version.
* @r	aises GetBasicCmIRPNotificationProfile when the system for some
*	reason cannot return the supported notifications and parameters.
* @r	eturns the list of all supported notifications and their supported
*	parameters for the specified version.
*/	
Manag	edGenericIRPConstDefs::MethodList
g	et_basicCm_IRP_notification_profile (
	in ManagedGenericIRPConstDefs::VersionNumber basicCm_IRP_version
)	
raise	s (
G	etBasicCmIRPNotificationProfile,
M	anagedGenericIRPSystem::OperationNotSupported,
M	
/**	
* Pe	rforms a containment search, using a SearchControl to
* CO	ntrol the search and the returned results.
*	
* Al	l MOs in the scope constitute a set that the filter works on.
* Th	e result BasicCmInformationIterator contains all matched MOs,
* wi	th the amount of detail specified in the SearchControl.
* Fo	r the special case when no managed objects are matched in
* İl	nd_managed_objects, the BasicCmInformationIterator will be returned.
* EX	eculing the next_basicCminformations in the
<u>Ва</u>	mpletion
*	
g@ *	arm baseObject The start MO in the containment tree.
* @p	arm searchControl the SearchControl to use.
* @p	arm requestedAttributes defines which attributes to get.
*	If this parameter is empty (""), all attributes shall
*	be returned. In this version this is the only supported semantics.
*	Note that this argument is only
*	relevant if ResultContents in the search control is
*	specifed to NAMES_AND_ATTRIBUTES.
*	
* @~	aises ManagedGenericIRDSystem: ValueNotSupported if a valid but
<u>יש "</u> מוו *	supported parameter value is passed F a the contents
* fi	eld in the searchcontrol parameter contains the value NAMES and
* t.h	e optional getContainment IS operation is not supported.
* @r	aises UndefinedMOException The MO does not exist.
* @r	aises IllegalDNFormatException The dn syntax string is
* ma	lformed.
* @r	aises IllegalScopeTypeException The ScopeType in scope contains
* an	illegal value.
* @r	aises IllegalScopeLevelException The scope level is negative

* (<0).
* @raises IllegalFilterFormatException The filter string is
* malformed
* @raises FilterComplexityLimit if the filter syntax is correct
* but the filter is the sempler to be presented by the IDD esempt
but the lifter is too complex to be processed by the inplagent.
* @see SearchControl
* @see BasicCmInformationIterator
*/
BasicCmInformationIterator find_managed_objects(
in BasicCMIRPConstDefs::DN baseObject,
in BasicCMIRPConstDefs::SearchControl searchControl,
in BasicCMIRPConstDefs::AttributeNameSet requestedAttributes)
FindManagedObjects
Finational geodesic fination is a second sec
ManagedGenericityPSystem: FarameterNotsupported,
ManagedGenericizPSystem: invalidParameter,
ManagedGenericIRPSystem::ValueNotSupported,
ManagedGenericIRPSystem::OperationNotSupported,
UndefinedMOException,
IllegalDNFormatException,
IllegalScopeTypeException,
IllegalScopeLevelException
IlegalFilterFormatException
* Performs the creation of a MO instance in the MIB maintained
* by the IRPAgent.
*
* @parm objectName: the distinguished name of the MO to create.
* @parm referenceObject: the distinguished name of a reference MO.
* @parm attributes: in input, initial attribute values for the MO to
* create; in output, actual attribute values of the created MO.
* @parm attributeErrors: errors, related to attributes, that caused the
* creation of the MO to fail
* evolves ManagadGanavisTDDGvetam::OnevoltionNetCurrented: The encyclica
* windses ManagedGenericikPSystemOperationNotSupported. The operation
<u> </u>
* @raises ManagedGenericIRPSystem::ParameterNotSupported: An optional
* parameter is not supported.
* @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
* parameter value has been provided.
* @raises UndefinedMOException: The MO does not exist.
* @raises IllegalDNFormatException: The DN syntax string is malformed.
* @raises DuplicateMO: A MO already exist with the same DN as the one
* to create
* @raises CreateNotAllowed: The creation of the MO is not allowed
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed.</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO.</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized.</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist.</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (in BasicCMIRPConstDefs::DN objectName, in BasicCMIRPConstDefs::DN referenceObject,</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (</pre>
<pre>* @raises CreateNotAllowed: The creation of the MO is not allowed. * @raises ObjectClassMismatch: The object class of the MO to create does * not match with the object class of the provided reference MO. * @raises NoSuchObjectClass: The class of the object to create is not * recognized. * @raises ParentObjectDoesNotExist: The parent MO instance of the * ManagedEntity specified to be created does not exist. */ void create_managed_object (in BasicCMIRPConstDefs::DN objectName, in BasicCMIRPConstDefs::MOAttributeSet attributes, out BasicCMIRPConstDefs::AttributeErrorSeq attributeErrors } raises (CreateManagedObject </pre>

<pre>ManagedGenericIRPSystem::OperationNotSupported,</pre>
——ManagedGenericIRPSystem::ParameterNotSupported,
<pre>ManagedGenericIRPSystem::InvalidParameter,</pre>
UndefinedMOException,
<pre>IllegalDNFormatException,</pre>
DuplicateMO,
CreateNotAllowed,
ObjectClassMismatch,
NoSuchObjectClass,
ParentObjectDoesNotExist);
/**
* Performs the deletion of one or more MO instances from the MIB
* maintained by the IRPAgent, using a SearchControl to control the
* instances to be deleted.
*
* All MOs in the scope constitute a set that the filter works on.
* All matched MOs will be deleted by this operation.
* The returned DeleteResultIterator is used to retrieve the DNs of the
* MOs deleted and the errors that may have occurred preventing deletion
* of some MOs.
* For the special case when no managed objects are matched in
* delete_managed_objects, the DeleteResultIterator will be returned.
* Executing the next_basicCmInformations in the DeleteResultIterator
* will return FALSE for completion.
*
* @parm baseObject: the start MO in the containment tree.
* @parm searchControl: the SearchControl to use; field "contents" has no
* meaning here and shall be ignored.
<pre>@returns: a DeleteResultIterator (see above).</pre>
*
* @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
* is not supported.
* @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
* parameter value has been provided.
* @raises UndefinedMOException: The MO does not exist.
* @raises IllegalDNFormatException: The DN syntax string is malformed.
* @raises IllegalScopeTypeException: The ScopeType in scope contains
* an illegal value.
* @raises IllegalScopeLevelException: The scope level is negative (<0).
* @raises IllegalFilterFormatException: The filter string is malformed.
* @raises FilterComplexityLimit: The filter syntax is correct,
* but the filter is too complex to be processed by the IRPAgent.
*/
DeleteResultIterator delete_managed_objects (
in BasicCMIRPConstDefs::DN baseObject,
in BasicCMIRPConstDefs::SearchControl searchControl
raises (
DeleteManagedObjects,
<pre>ManagedGenericIRPSystem::OperationNotSupported,</pre>
——ManagedGenericIRPSystem::InvalidParameter,
UndefinedMOException,
<pre>IllegalDNFormatException,</pre>
UndefinedScopeException,
IllegalScopeTypeException,
IllegalScopeLevelException,
IllegalFilterFormatException,
FilterComplexityLimit);
/**
* Performs the modification of MO attributes. One or more MOs attributes
* may be modified according to a SearchControl

*
* All MOs in the scope constitute a set that the filter works on.
* All matched MOs will have their attributes modified by this operation.
* The returned ModifyResultIterator is used to retrieve the DNs of the
* modified MOs together with the values of the modified attributes, and
* the errors that may have occurred preventing modification of some
* attributes.
* For the special case when no managed objects are matched in
* modify managed objects, the ModifyResultIterator will be returned.
* Executing the next basicCmInformations in the ModifyResultIterator
* will return FALSE for completion.
*
* @parm baseObject: the start MO in the containment tree.
* @parm searchControl: the SearchControl to use; field "contents" has no
meaning here and shall be ignored.
* @parm modifications: the values for the attributes to modify and
the way those values are to be applied to the attributes.
@returns: a ModifyResultIterator (see above).
*
* @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
<pre>* is not supported</pre>
* @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
<pre>* parameter value has been provided</pre>
* @raises UndefinedMOException: The MO does not exist.
* @raises IllegalDNFormatException: The DN syntax string is malformed.
* @raises IllegalScopeTypeException: The ScopeType in scope contains
* an illegal value.
* @raises IllegalScopeLevelException: The scope level is negative (<0).
* @raises IllegalFilterFormatException: The filter string is malformed.
* @raises FilterComplexityLimit: The filter syntax is correct,
* but the filter is too complex to be processed by the IRPAgent.
<u>*/</u>
ModifyResulfIterator modify_managed_objects (
in BasicCMIRPConstDefs::DN baseObject,
in BasicCMIRPConstDefs::SearchControl searchControl,
in BasicCMIRPConstDers::AttributeModificationSet modifications
MonifyManagedobjects,
ManagedGeneridIPPSystem::OperationNotSupported,
ManagedGenerictRPSystemInvaridParameter,
UllegalDNFormatFxception
UndefinedScopeException
IllegalScopeLevelException
IllegalFilterFormatException
};
$_{i}$

#endif _BASICCMIRPSYSTEM_IDL_

End of change in Clauses A End of document