INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.812

Appendix I

(03/99)

SERIES Q: SWITCHING AND SIGNALLING

Specifications of Signalling System No. 7 – Q3 interface

Upper layer protocol profiles for the Q3 and X interfaces

Appendix I – Guidance on using allomorphic management

ITU-T Recommendation Q.812 – Appendix I

(Previously CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
General	Q.700
Message transfer part (MTP)	Q.701–Q.709
Signalling connection control part (SCCP)	Q.711–Q.719
Telephone user part (TUP)	Q.720–Q.729
ISDN supplementary services	Q.730–Q.739
Data user part	Q.740–Q.749
Signalling System No. 7 management	Q.750–Q.759
ISDN user part	Q.760–Q.769
Transaction capabilities application part	Q.770–Q.779
Test specification	Q.780–Q.799
Q3 interface	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION Q.812

UPPER LAYER PROTOCOL PROFILES FOR THE Q3 AND X INTERFACES

APPENDIX I

Guidance on using allomorphic management

Source

Appendix I to ITU-T Recommendation Q.812, was prepared by ITU-T Study Group 4 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 26th of March 1999.

CONTENTS

Recommendation Q.812

UPPER LAYER PROTOCOL PROFILES FOR THE Q3 AND X INTERFACES

APPENDIX I

Guidance on using allomorphic management

(Geneva, 1999)

I.1 Introduction

This appendix seeks to provide guidance to the developers of CMIP managers and agents in the use of allomorphism. Allomorphism is a powerful concept that is of increasing value as TMNs are implemented. Allomorphism can be used to address the issue of how to add new capabilities to existing TMN manager and agent implementations. As requirements evolve and models are extended to satisfy those requirements, software in the managing system that takes advantage of allomorphism can be written in such a way that it does not require to be rewritten until the new features in the model are needed.

This appendix attempts to clarify how to cope with allomorphic behaviour in implementations of both manager and agent systems. It clarifies the description of allomorphism found in Recommendation X.720. In particular, managers must be aware of allomorphism to benefit from it. Even if a manager does not plan to use allomorphism, it should have a minimum ability to interface with agents that do implement allomorphism. For example, the manager must support the allomorphs attribute and have the ability to construct filters using allomorphs versus the actual class in the objectClass attribute.

This appendix discusses the issues related to allomorphism for each CMIP operation from both the manager and agent perspective. It then discusses the issues related to CMIP Notifications, again from both the manager and agent perspective. It then provides some protocol stack and implementation considerations. It concludes by answering some frequently asked questions about allomorphism.

In the following discussion, the phrase "if agent supports allomorphism" is used. This should be equated to "if agent supports allomorphism for a specific instance" because it is possible that an agent may support revisions for some object classes and not others. Strictly speaking, even two instances of a class may be different in the support for allomorphism, however, this is considered as an extreme and rare implementation for this discussion. The same is also true for manager side where a specific release of a manager system may recognize multiple definitions for some basic classes and not for others. The decisions to include the different versions are dictated by business objectives (which are out the scope of this appendix).

I.1.1 Overview

Allomorphism is the ability of a managed object that is an instance of a given managed object class to be managed as a member of one or more other managed object classes. Allomorphism allows instances of one managed object class – referred to as the extended class to represent instances of another managed object class – the allomorphic class.

When an extended class is instantiated, the actual class (see Recommendation X.720) of the object stored in the objectClass attribute is the extended class. It is extended with regard to another managed object class, its compatible managed object class. The actual class is that class of which a managed object is an instance. An allomorphic class of a managed object is a managed object class other than the managed object's actual class; however it can be managed as an instance of that class. A managed object may be allomorphic to one or more of the compatible classes (i.e. instances of the extended class can be managed as instances of compatible managed object class). In other words the terms "allomorphic class" and "compatible managed object class" can be used as synonyms. When an agent creates a managed object that supports allomorphism, the allomorphicPackage (defined as a Conditional Package in the top class in Recommendation X.721) is included. The package contains the allomorphs attribute. This GET-Only attribute is set-valued and contains the object identifiers of the classes that this object can represent (is allomorphic with). The objectClass attribute has a value of the actual class used in creating this instance.

The basic idea behind allomorphism is that the extended class supports all the capabilities of the classes it is allomorphic with. It may also support additional capabilities. The extended class may be a subclass of the classes it is allomorphic with but this is not required. In all respects the extended class behaves as the class it actually is. This may mean that the manager could receive information that is not in the allomorphic class. For example, if the extended class has new attributes, a Get all operation from the manager will return values for these. Dealing with this type of issue requires the manager to be aware that allomorphic management is being used. Recommendation X.726 defined Managed Object Conformance tables for use by both agent and manager implementations. Use of these tables to identify the list of allomorphs if supported is recommended to determine the levels of interoperability.

The various interactions between manager and agents using allomorphic management are discussed in the following clauses. Recommendation X.720 also discusses limited interoperability when compatible rules are not completely satisfied. This appendix addresses only the scenarios where compatible rules defined according to Recommendation X.720 to support allomorphism are met (See 5.2.3.2/X.720).

I.2 CMIP operations

This clause discusses allomorphism in relation to the CMIP Operations, m-Create, m-Get, m-Set, m-Action, and m-Delete. No impact of allomorphism on the m-Cancel-Get operation has been identified.

I.2.1 Creating managed objects

A managed object is created either by the manager issuing an explicit create request on the interface or by automatic create within the agent system. Each case is discussed below. Careful selection of the naming attribute and structure is required. The naming issues are discussed below in a separate subclause.

I.2.1.1 Explicit creation – Manager role

Case 1: The manager issues a CMIP create request providing in the managed object class a value and a set of attribute values appropriate for that class. The manager supplies the actual name of the new object. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, c. If the response is as defined in case c, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 2: The manager issues a CMIP create request providing in the managed object class a value and a set of attribute values appropriate for that class. The manager supplies the name binding attribute value. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, d. If the response is as defined in case d, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 3: The manager specifies the class without a specific name in the instance field or a value for the name binding attribute. The resulting action in the agent is one of those defined in I.2.1.2 Explicit creation – Agent role, cases a, b, e. If the response is as defined in case e, the manager must be able to ignore unknown attributes included as a result of creating an extended class.

Case 4: The manager specifies a class and requests it to be a copy of another object, i.e. with reference object. Depending on the value of the class, any of the above three cases are possible.

I.2.1.2 Explicit creation – Agent role

Case a: Agent recognizes the managed object class in the request and supports it as actual class. In this case the requested managed object class is created and allomorphism is not involved. The create succeeds or fails depending on the conditions associated with the behaviour and the attribute values supplied by the manager. The agent uses either the name supplied in case 1 above or assigns a name (using either the name binding rule in case 2 or internally generated based on the schema definition).

Case b: Agent does not support the requested class either as an actual class supported or as an allomorph. The managed object class provided in the request is not recognized. The create request is rejected with the error "no such object class". This is a normal failure to create an unknown class.

Case c: Agent supports allomorphism and it creates a class that is an extended class with the name supplied by the manager. This assumes the name provided by the manager follows the structure rules (name binding) for the extended class [same or extended superior class, same Relative Distinguished Name (RDN)]. If this condition is not met, then the create will fail. If the create succeeds, the agent responds with the value for the actual class in the managed object class field and all the attributes appropriate for the actual class. The object is created according to the behaviour of the extended class. If the agent system is providing the interoperability (see 5.2.3.1/X.720), the agent includes the attribute allomorphs with the value of the class provided in the create request. If manager system is providing the interoperability (see 5.2.3.2/X.720), the manager may query allomorphs attribute and determine that requested object class is allomorphic to the actual class.

Case d: Agent supports allomorphism and it creates a class that is an extended class with the name binding supplied by the manager. This assumes that the name binding supplied is valid for the extended class. This is often true if the extended class is a subclass and the GDMO includes for name binding "WITH SUBCLASSES" clause. The response when create succeeds is the same as for case c. If the name binding is not valid for the extended class (for example, additional behaviour may be included in a different name binding for the extended class even if the naming attribute and structure are the same) the create will fail. Note that if the manager receives invalid value error for the name binding attribute without further information, this will not help in resolving the problem. From the manager's perspective the request is for a class and the name binding is valid. Because the extended class requires a different behaviour, the agent cannot use that name binding. This is why the recommendation is not to supply the name binding attribute in the create request.

Case e: The agent supports allomorphism and creates a class with an appropriate name selected by the agent (for case 3 of I.2.1.1: Explicit creation – Manager role). The assigned name may or may not be understood by the manager depending on chosen name binding definition. Other information in the response is the same as in case c.

For all the above cases, when the manager does not supply a value for an attribute and a default exists, the value chosen for that attribute is according to the created object class. In cases c, d, and e this can result in the manager receiving values for attributes that exist only in the extended class. The manager may not understand these types and should be able to ignore these types without disrupting the association. Note that the manager may wish to take additional management action to record that it is encountering allomorphic agents (e.g. log information not understood).

In addition to default values of attributes, constraints imposed on initial value, permitted and required values may have differences between the allomorphic class and the extended class. See I.4 for further discussions.

I.2.1.3 Summary

The table below summarizes the various cases for explicit creation discussed from manager role and agent role depending on whether allomorphism is supported or not.

Manager role	**Agent role with allomorphism**	
	Supported	Not supported
Case 1	Cases a, b, c	Cases a, b
Case 2	Cases a, b, d	Cases a, b
Case 3	Cases a, b, e	Cases a, b
Case 4	Cases a, b, c, d, e	Cases a, b

I.2.1.4 Automatic creation – Agent role

The agent may create a managed object internally and inform the managers of the creation by the object creation notification.

Case 1: Assume that the agent has implemented the extended class and allomorphic behaviour. In addition to all the attributes appropriate for the created class, the allomorphs attribute containing all the allomorphic (compatible) object classes is included in the created managed object. The agent sends a create notification for the new object.

In this case, the agent then sends a notification to all the managers using the object creation notification containing the actual class. It includes all the attributes of the class referenced in the managed object class field of the create notification. This includes the allomorphs attribute.

Note that the default values used are consistent with the created class.

Case 2: The agent only implements the extended class and does not exhibit allomorphism for the compatible classes. In this case the object creation notification contains only the information pertaining to the extended class and the allomorphs attribute is not present. For such an environment, it is recommended that the managing system provide for interoperability independent of whether the additional features are required or not. Rejecting the notification will not be useful for a practical TMN environment.

I.2.1.5 Automatic creation – Manager role

The manager receives the object creation notification with the allomorphs attribute and an extended class in the managed object class field. Depending on whether the manager supports allomorphism or not, the cases described in a to d are applicable for case 1 above.

Case a: The manager knows about the extended class and the allomorphs attribute is not needed because the manager does not have to perform allomorphic management. All the characteristics associated with recognizing or not of the attribute identifiers and values are the same when no allomorphism is used.

Case b: The manager does not recognize the managed class value in the notification. If the manager understands the allomorphs attribute identifier, then the manager, before ignoring the notification as unrecognized information, should examine the allomorphs attribute value to determine if it can manage using one of the values in this attribute. This implies that at least one object class value in the allomorph attribute is recognized by the manager. The manager must ignore all the attributes not recognized as belonging to the allomorphic class.

Case c: The manager does not recognize the managed object class value and it has not implemented the ability to recognize the allomorphs attribute identifier. In this case the manager cannot manage this auto created object. If the creation is sent with a confirmation, an error response may be provided by the manager to say unknown object.

Case d: The manager does not recognize the managed object class value in the notification. The manager understands the allomorphs attribute; however it does not recognize any of the classes in the allomorphs attribute. In this case the result is the same as case c. Managing the auto-created object by that manager is not possible.

The agent sends a notification using the actual class without the allomorphs attribute (agent does not allomorphism). This corresponds to case a above.

Case e: The manager understands the extended class and the behaviour is as in case a above. The absence of allomorphs attribute is not relevant and the manager can manage the auto created object.

Case f: The manager does not recognize the extended class. The manager will not be able to manage this object, given that the allomorphs attribute is not supported by the agent. This is true irrespective of whether the manager supports allomorphism or not.

I.2.1.6 Summary for AutoCreation

The table below summarizes the relationship of the manager and agent cases.

Agent case	Manager case
1	a, b, c, d
2	e, f

I.2.2 Get operation

The Get operation may be issued with an explicit set of attribute identifiers, an empty list or a missing list. The two cases (empty list and missing list are treated the same) are further separated in terms of whether the managed object class in the request is the actual class or allomorphic class for the agent. Note that there is a special object identifier (42) that the manager may use in the request to refer to the actual class of the managed object without having to specify the actual class.

I.2.2.1 Manager role

Case 1: Manager issues a get request with a class that is not the actual class of the object but one of the allomorphs supported by the agent. The list of attribute identifiers included are appropriate for this allomorphic class. The response received is according to one of cases a, b, c below. If interoperability is supported by the manager, then receiving a successful response in case b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues a get request with a class which is either the actual class or the special object identifier which implies the actual class. The list of attributes requested may or may not be appropriate for that class. This is because of either conditionally or by using the special object identifier, the manager may include attributes that are not available for the implemented class. The response received is according to case d below.

Case 3: Manager requests specifying a class, name of a managed object and no attribute list. The response received depends on the value in the class, whether allomorphism is supported by agent and the method of interoperability. The response received is one of cases e, f, g given below. If the response in f or g contains attributes, the manager does not understand, they are ignored by the manager.

Case 4: Manager requests specifying either a class supported by the agent and name of a managed object or the special object identifier discussed above and a name. It does not include the attribute list. The response received is according to case h below. The manager should ignore attributes that are not recognized when the class in the response is different from what the manager recognizes (as a result of using the special object identifier).

I.2.2.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent responds with the values of the attributes requested and the class as in the request may or may not be included. If the class is included, it is recommended that the value of the actual class be used in the response. This approach provides for uniform and consistent response irrespective of single object request or multiple object request using scoping. It is also expected that the managed object class parameter in a CMIP response corresponds to the value of the objectClass attribute.

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may either return an error ("no such object class" or "class instance conflict") or respond with the values of the attribute (agent provided interoperability). The latter behaviour is recommended. The class field may be left empty or populated with the actual class. If both class and name are not recognized, an error response is generated (no such object instance or no such object class).

Case c: Agent does not support allomorphism and does not understand the value of the class or the name in the request. An error "no such object class or no such object instance" is returned.

Case d: The agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class or the simple case where both the manager and agent knows about only one class). Agent responds with attribute values (includes errors if the attributes requested are not suitable for that class or have not been implemented as a result of conditionality). The class field may be omitted or the actual class is included.

Case e: Agent recognizes the class and name in the request (irrespective of allomorphism is supported or not) and returns all the attribute identifiers and values for that object. Class and name may be omitted in the response.

Case f: Agent supports allomorphism. The value of the class in the request does not match the value of the actual class even though the name corresponds to one of the objects contained in the system. The class corresponds to a value in the allomorphs attribute. If the agent provides interoperability, the agent responds with only the value of the attributes appropriate for the requested class. If the value of the class is included in the response (it is not required to include either the class or name), then the actual class is used (see explanation above on the same topic). If the manager provides interoperability, the agent returns all of the attributes included in the object. If the value of the class in the request does not correspond to any of the values in the allomorphs attribute, then the agent returns an error "no such object class" to the manager. Note that in order to provide for interoperability, it is recommended that both agent and manager provide some capabilities: the agent supporting allomorphism and the manager ignoring unknown information.

Case g: Agent does not support allomorphism. The requested class is not recognized but an object with that name is available. The agent may either respond with an error "no such object class" or all the attributes relevant to that class. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case. This provides the manager with the information that the actual class is an extended class of the compatible class the manager understands.

Case h: Requested class corresponds to the actual class in the agent or the actual class is used because the request contained the special object identifier value. Irrespective of whether allomorphism is supported or not, all the attribute values corresponding to the implemented actual class for that object are returned assuming that the agent recognizes the name. The class and name fields may or may not be in the response. It is, however, recommended that the actual class and name be included in this case when the special object identifier is used in the request. If the name is not recognized, then "no such object instance" error is returned.

I.2.2.3 Summary GET operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d
3	e, f, g
4	h

I.2.3 Set operation

The Set operation may be issued with different operators. The replace may be specified either with a specific value or "set to default". The default value for an attribute may depend on the actual class. An attribute may be specified in one class with a set of permitted values and a set of required values. The required values must be a subset or an equal set of the permitted values. The extended class shall not increase the permitted values but may remove some as long as these are not in the required set of values (see Figure 1 in I.4). Not supporting a permitted value is permitted either in the extended or compatible class as long as this value is not in the required list. Thus, with allomorphic management, when a permitted value for the allomorph is given and this is not included in the extended class, it can be rejected without violating allomorphic behaviour (guaranteed to support all values in the compatible class; however if the value is not in the permitted set of compatible class, this will not be supported as the list cannot be extended).

I.2.3.1 Manager role

Case 1: Manager issues a set request with a class, name and the value(s) for the attributes (replace/add or remove operator). The class of the object is not the actual class in the agent but a compatible class. If a response is received (only if the request was confirmed), one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in case b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues a set request with a class which is either the actual class or the special object identifier which implies the actual class. The value(s) for the attributes (replace/add or remove operator) may or may not be appropriate for that instance. This is because of conditionality or by using the special object identifier, the manager may be providing values appropriate for the compatible class. If a response is received (only if the request was confirmed), case d below is valid.

Case 3: Manager requests specifying a class, name and a replace with default for one or more attributes. The response (if received) depends on the value in the class, whether allomorphism is supported by agent and the method of interoperability. It is one of e, f or g below. The response received may indicate default values for the attributes that is different from those associated with the requested class. The manager should recognize these values as the result of actual class in the agent being different.

Case 4: Manager requests specifying either a class supported by the agent and name or the special object identifier discussed above and name and a replace with default for one or more attributes. The response, if received, is according to case h below.

I.2.3.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent performs the modification for those attributes (assuming the attribute exists and the value provided is valid). If the request is confirmed, then the agent responds with either an acknowledgment or the modified values. In the latter case, the value for the class field (if present) is the actual class (see the rationale above for Get). If the attributes or values provided in the request are invalid, then an error or partial success (set list error) is returned.

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may not perform the requested modification based on not recognizing the class. If the request was unconfirmed, then the manager has no knowledge of the result unless it issues a get operation. Consider the case where a response is required. Depending on whether the agent performed the operation (successfully or otherwise), it may return one of the following: an error ("no such object class" or "class instance conflict"); confirmation indicating success, error with partial success. The class and name fields are not required to be in the response. If present, it is recommended to include the actual class to inform the manager of the implemented class. If both class and name are not recognized, an error response is generated (no such object instance or no such object class).

Case c: Agent does not support allomorphism and does not understand the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class or the simple case where both the manager and agent knows about only one class). The agent performs the modification for those attributes (assuming the attribute exists and the value provided is valid). If the request is confirmed, then agent responds with either an acknowledgment or the modified values. In the latter case, the class and name fields are not required to be in the response. If present, the value for the class field is the actual class (same rationale as in Get). If the attributes or values provided in the request are invalid, then an error or partial success (set list error) is returned.

Case e: Agent recognizes the class and name in the request (irrespective of allomorphism is supported or not) and performs the operation. The response, if required may be an acknowledgment, an error because there is no default defined or the modified value (the default specified for that class).

Case f: Agent supports allomorphism. The value of the class in the request does not match the value of the actual class even though the name corresponds to one of the objects contained in the system. The class corresponds to a value in the allomorphs attribute. If the agent provides interoperability, the agent either performs the modification according to the default value for the actual class or detects an error (e.g. no default value defined for some of the attributes). It responds with either an acknowledgment or the assigned default value or an error with partial success. It is not required to include the class and name fields in the response. If the value of the class is included in the response, then it is the actual class (see rationale in Get). If the value of the class in the request does not correspond to any of the values in the allomorphs attribute, then the agent returns an error "no such object class" to the manager.

Case g: Agent does not support allomorphism. The requested class is not recognized but an object with that name is available. The agent may either perform the operation replacing with defaults appropriate for the actual class or reject the request. If the request was confirmed and the agent rejects the request, then the agent responds with either "no such object class" or "class instance conflict" error. If it performs the operation successfully, then either an acknowledgment or a success with the modified values are returned. The returned values are appropriate for the actual class. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case. This provides the manager with the information that the actual class is an extended class of the compatible class the manager understands (manager provided interoperability). If the modification is performed with partial success, then an error is sent. The use of the class field is the same as for the success case.

Case h: Requested class corresponds to the actual class in the agent or the actual class is used because the request contained the special object identifier value. Irrespective of whether allomorphism is supported or not, the agent replaces with default all the values corresponding to the attribute identifiers in the request (assuming all the attributes in the request are supported by the agent). If it performs the operation successfully, then either an acknowledgment or a success with the modified values is returned. Even though the response is not required to include the class and instance for single object request, it is recommended that the actual class and name be included in this case when the special object identifier is used in the request. If the name is not recognized, then "no such object instance" error is returned.

I.2.3.3 Summary SET operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	**Agent case**
1	a, b, c
2	d
3	e, f, g
4	h

I.2.4 Action operation

The action operation for any specific may include argument and responses both with mandatory and optional parameters. If there are required parameters in the argument for the actual class that are not part of the compatible class, then they must have defaults associated with them. Even though Recommendation X.720 permits the addition of required parameters in the action information, it is not possible to specify this using the template notation without creating a new action. However, the requirement may be specified via the behaviour. This is because only the labels of the parameters can be used to augment an action specification. This implies the original action has a field that is ANY DEFINED BY (or information object class in Recommendation X.681) and is augmented with parameter template label in another class (or by creating an information object). If the required fields are to be added to an action, the only approach available is to define a new action, which has a different registration. In other words, the templates do not support deriving an action from another action by adding new fields that are mandatory in a formal ASN.1 specification.

I.2.4.1 Manager role

Case 1: Manager issues an action request with a class, name and the value(s) for the parameters of the action argument (if present). The class of the object is not the actual class in the agent but a compatible class. If a response is received (only if the action was defined as confirmed), one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in cases a and b with a different class and additional fields in the action reply will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP).

Case 2: Manager issues an action request with a class which is either the actual class or the special object identifier which implies the use of actual class. Not all the fields included may or may not be appropriate for that class. If a response is received (only if the request was confirmed), case d below is valid.

I.2.4.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent performs the action according to its actual class using the parameters supplied in the request. Note that if there are any additional fields required for performing the action, then default values must be available as they will not be supplied by the manager. If the request is confirmed, the response from the agent depends on the following: if the action was performed successfully and if agent provides for interoperability. The response is not required to include the class and name in this case where a single object is referenced. If the operation does not succeed, an error is returned. The agent may include the actual class to inform the manager of the implemented class or the requested class (allomorphic class). If the action succeeds, the agent responds with either an acknowledgment (a confirmation that the action was performed successfully because the action definition does not include any fields for the response) or the action response with appropriate fields. The agent may choose one of the two following methods to respond. If the agent provides for interoperability, it may include only the fields appropriate for the requested class and not the additions for the actual class. In this case the value for the class field can be omitted. In the second approach, manager provided interoperability is assumed. The response may include additional fields that were not in the action for the class in the request (new parameter templates may have been included for the extension field or application of extensibility rules in ASN.1). It is recommended to include in the class field the actual class to let the manager be aware of the implemented class.

Case b: Agent does not support allomorphism but recognizes the name of the managed object and the action is valid for its actual class. The agent may not perform the requested action based on not recognizing the class in the request (different from the actual class). If the action definition indicates unconfirmed, then the manager has no knowledge of whether the action was successful or not. Depending on the action type, the effect of that action may be deduced later (for example by doing a get operation). Consider the case where the action is confirmed. Depending on whether the agent performed the action (successfully or otherwise) it may return one of the following: an error ("no such object class" or "class instance conflict"); confirmation indicating success, specific error (if any) defined for that action or a generic CMIP error. The action is performed according to the actual class.

The class and name fields are not required to be in the response. If present, it is recommended to include the actual class to inform the manager of the implemented class. If both class and name are not recognized, an error response is generated (no such object instance).

Case c: Agent does not support allomorphism and does not recognize the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class thus supporting manager provided interoperability or the simple case where both the manager and agent knows about only one class). The agent performs action according to its actual class irrespective of whether the request contained the allomorphic class or the special object identifier. The result of the action may be successful or an error. If the action was not defined as confirmed no response is generated. If successful or an error occurs in performing the action, then appropriate result or error response is issued. If the action succeeds the response is generated according to the definition for the actual class. If the special object identifier is used, it is recommended to include the class value for the actual class even though the class and name fields are not required for the single object case.

I.2.4.3 Summary of ACTION operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d

I.2.5 Delete operation

The delete operation is defined with two options: deletes contained objects and no delete allowed unless all contained objects are deleted. The following must be noted for delete operation. Irrespective of the class, it is the name that must be recognized because two instances of the same class may have different names and/or behaviour (based on the name binding used to instantiate the object).

I.2.5.1 Manager role

Case 1: Manager issues a delete request with a class and a name. The class of the object is not the actual class in the agent but a compatible class. For response one of cases a, b, c below is valid. If interoperability is supported by the manager, then receiving a successful response in cases a and b with a different class will be recognized as a valid response. Otherwise, the manager will reject the response (a ROSE reject and not from CMIP). As noted earlier, sending a reject is not useful (thus not recommended) and the manager should provide for some level of interoperability.

Case 2: Manager issues a delete request with a class which is either the actual class or the special object identifier which implies the use of actual class. The response shown in case d below is valid.

I.2.5.2 Agent role

Case a: Agent supports allomorphism and recognizes the name of the managed object. The managed object class in the request matches one of the values in the allomorphs attribute. The agent deletes the object assuming the conditions for deletion is acceptable according the name binding used for that instance. If the deletion is not performed, an error is generated. If the deletion succeeds, the agent responds with either an acknowledgment (a confirmation that the deletion was performed successfully). The response is not required to include the class and name in this case where a single object is referenced. It is recommended to include in the class field the actual class to let the manager be aware of the implemented class (this may not be useful for the manager since the object is deleted unlike the previous operations).

Case b: Agent does not support allomorphism but recognizes the name of the managed object. The agent may not delete the object based on not recognizing the class ("no such object class" or "class instance conflict"). If the agent performs the deletion based on the object name (assuming other conditions for deletion are met), then an acknowledgment is returned. If deletion is not possible (conditions are not met), then an error is returned. In either case, it is not required to include the class and name of the object. It is recommended to include the actual class (this may not be useful for the manager since the object is deleted unlike the previous operations).

Case c: Agent does not support allomorphism and does not recognize the name in the request. An error "no such object class" or "no such object instance" is returned.

Case d: Agent recognizes the class and the name irrespective of whether it supports allomorphism or not (this corresponds to case 2 where the manager requests using a class that is a compatible class and not the extended class even though the manager knows about the extended class thus supporting manager provided interoperability or the simple case where both the manager and agent knows about only one class). The agent deletes the object assuming the conditions for deletion is acceptable according the name binding used for that instance. If the deletion is not performed an error is generated. If the deletion succeeds, the agent responds with either an acknowledgment (a confirmation that the deletion was performed successfully). The response is not required to include the class and name in this case where a single object is referenced. It is recommended to include in the class field the actual class (if the special object identifier was used; otherwise the manager and agent has the same understanding of the class value) to let the manager be aware of the implemented class (this may not be useful for the manager since the object is deleted unlike the previous operations).

I.2.5.3 Summary of DELETE operation

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
1	a, b, c
2	d

I.3 CMIP notification

Notifications are much simpler than the operations mentioned in the previous clause. The notification may be sent in the confirmed or unconfirmed mode. If unconfirmed, the manager may ignore what it receives because any of the following is not recognized: class, name, event type and any field of the event information. The use of RO-Reject is always possible but this is sent at ROSE level and not recognized by CMIP. However, sending a reject does not provide for TMN interoperability.

I.3.1 Manager role

Case a: The manager understands the class, name, notification type and some of the parameters in event information. The manager ignores the unknown parameters.

Case b: The manager does not provide for interoperability. It does not understand the class. The manager should ignore the notification. The manager may choose to send a reject.

Case c: The manager understands the name, notification type and some of the parameters in event information. The manager does not understand the class. The manager may ignore the notification or determine that the class is an extended class and ignores the unknown parameters (if manager provides for interoperability). Otherwise the manager may send a reject.

Case d: The manager does not recognize class, and the name (in this case not recognizing other parameters of the notification may not matter). The manager should ignore the notification.

I.3.2 Agent role

Case 1: Agent supports allomorphism. If the agent provides for interoperability, the notification may include in the class field a value from the allomorphs attribute and all the event information (irrespective of whether the parameters are applicable to the allomorphic class or not). Normally, it is expected that the agent will use the actual class because in general the agent has no knowledge which of the allomorphs should be used in the event report. If the notification is confirmed, cases a, b or c above apply.

Case 2: Agent does not support allomorphism. The agent issues the notification using the actual class and the relevant parameters for that notification. Cases a, b, c or d above are valid.

I.3.3 Summary for NOTIFICATION

The table below summarizes the relationship of the manager and agent cases.

Manager case	Agent case
a	1, 2
b	1, 2
c	1, 2
d	2

I.4 Implementation issues

I.4.1 Protocol stack related

At several places in the discussion of allomorphism it was required that the manager be able to ignore ASN.1 syntaxes (either from attributes only in the extended class or from notifications only in the extended class). Generally, these syntaxes are not ones that have been implemented in the manager (since the manager is attempting to do allomorphic management). In order for the manager to successfully ignore these syntaxes, they must be allowed to pass through the protocol stack without disrupting the association. This is supported by the presentation layer protocols. The presentation layer normally has a requirement to abort associations if an unknown PDU is received (See 6.4.4.3/X.226). However, 7.5/X.711 (CMIP) states that CMIP resolves all ANY DEFINED BY OID syntaxes. To quote 7.5/X.711:

> The corresponding ASN.1 object descriptor value shall be "CMIP-PCI".
>
> This abstract syntax is defined to include all data types resolved by the ANY DEFINED BY X productions, in which X is of type OBJECT IDENTIFIER.

I.4.2 Permitted and required values

The relationship of the values of attributes between the allomorphic classes and extended classes is similar to the relationship between super classes and their subclasses. The Required Values of the allomorphic class must be a subset of the Required Values of the extended class. In turn the Required Values of the extended class must be a subset of the Permitted Values of the allomorphic class.

Finally, the Permitted Values of the extended class must be a subset of the Permitted Values of the extended class. These relationships are illustrated in Figure I.1, below.

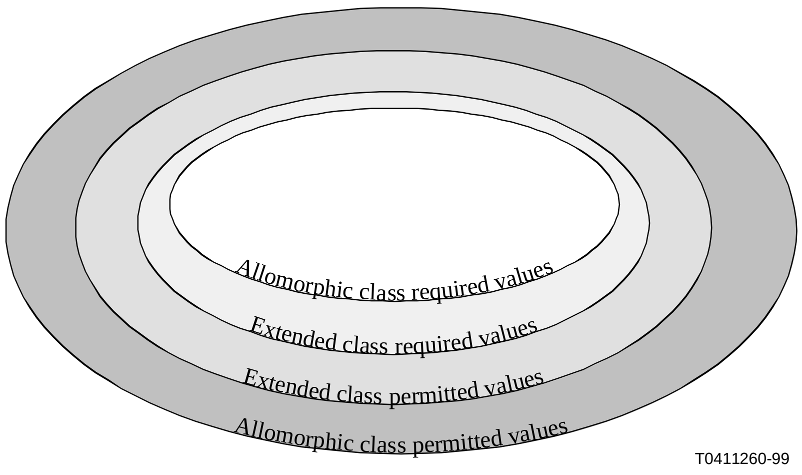
Allomorphic class required values
Extended class required values
Extended class permitted values
Allomorphic class permitted values
T0411260-99

Figure I.1/Q.812 – Relationship of permitted and required values

NOTE – Even though the above rules are to be adhered to in developing the model, it may be difficult in practice. An example is the multiport circuitpack and circuitpack defined in Recommendation M.3100. The permitted values of the availability status restricted to only one value whereas practical experience showed other values are required. Hence this restriction was removed in multiport circuitpack. However, the new version of circuitpack could not be subclassed from circuitpack as the permitted values cannot be extended and therefore strictly speaking not allomorphic. Even though the instance of an extended class has a value for an attribute outside the permitted range for the compatible class, the managing system may provide some level of interoperability. It should be able to encode the ASN.1 syntax of the attribute.

I.4.3 Initial value

A specification of a managed object class may include initial values for zero or more attributes. Unlike default values, with initial values, create request will fail if the value provided in the request is different from the specified initial value. When the value is not present, then the agent supplies the specified initial value. An extended class may specify an initial value for a given attribute different from that for the compatible class. When the agent creates the managed object, the initial value appropriate for the actual class will be used. Thus, similar to the attributes with default values discussed in I.2.1.2, the manager may receive values for attributes that are different from the ones associated with the object class in the request. It is recommended when initial values are defined for an attribute, the manager does not supply the value in the create request. This will avoid the possible rejection of the request because the supplied initial value is not appropriate for the actual class.

I.4.4 Filtering on single object

When a filtered request is made, assuming the agent has recognized the request (using conditions stated above), the following cases arise:

Case 1: The filter has all the attributes it recognizes and has been implemented for the object. The filter operation is not impacted.

Case 2: The filter has attributes that are not implemented by the agent for that object either because these are conditional or because they are for the new version of the object. The condition being checked for any attribute is equivalent to (attribute exists and the value meets the stated condition). That part of the filter should evaluate to true to make the agents simple irrespective of the implemented class. If the attribute specified in the filter is objectClass and the value to be compared is that of the compatible class, then objects with extended class will not meet the criteria. If the manager requires objects belonging to both the extended and compatible classes be selected, then the filter should include OR{equal{objectClass, x}, nonullIntersection {{x}, allomorphs}}.

I.4.5 Scoping only

The manager requests operations by providing a base object and scope level.

Case 1: The base object is the class implemented by the agent:

– Agent does not support allomorphism and responds with the actual class for the selected objects (irrespective of whether it has implemented to the extended definition it understands and implemented only to one definition). The manager may receive responses for objects with unrecognized values for the class field (because the manager does not recognize the new schema). The manager may provide for limited interoperability based on the name and other characteristics it recognizes.

– Agent supports allomorphism and has implemented to extended definitions within the selected scope. It performs the operation according to the actual classes of the objects in the scope. The response uses the actual class in the managed object class because the agent may not be aware of which one or more of the values in the allomorphs attribute the manager can recognize and the details relevant to that class (attributes, action result, etc., as noted above). Even if it supports allomorphism, it is simpler to respond in this case using the actual class and the properties appropriate for that class. The manager may provide for interoperability if it understands both versions of the definitions or limited interoperability (only one version is recognized). (If the manager supports allomorphism it may be useful for the scoped GET to request that the value of the allomorphs attribute be returned.)

Case 2: The base object is a class different from the class in the request but the name is recognized (the class is either a newer definition than what is in the request or older definition):

– Agent does not support allomorphism and implements to an older definition. It may reject the request because class is not recognized or it may use the name and respond with objects in the scope. If the manager receives "no such object class" or "class instance conflict" it may resend it with the appropriate class for the base object. This is possible only if the manager provides for interoperability and has knowledge of the version supported by the agent. It is possible the agent identifies the base object from the name (irrespective of the class), selects the objects in the scope and responds to the manager using the actual class. The manager should ignore information that it does not recognize if it provides for interoperability.

– Agent supports allomorphism and determines if the base object class is an allomorph. If this is true then it performs the operations on the selected objects (using the actual class) and responds using the actual class (manager provided interoperability).

In both the above cases, if the base object is not identified, an error is returned.

Case 3: Though unlikely, it is possible for the agent to provide for interoperability if it has the knowledge of the versions supported by the manager(s). In this case, the response may be customized to the specific manager's knowledge.

I.4.6 Scoping and filtering

When both scope and filter are in the request, anyone of the three cases discussed above is to be considered. For each case that results in selecting the objects by applying scope, the filter discussions in I.4.4, "Filtering on single object" are applied. No additional behaviour is required.

I.4.7 Naming

As noted earlier, Recommendation X.720 defines allomorphism as a property of the managed object. In principle, it is not required for two classes (compatible and extended) to be related by inheritance in order to exhibit allomorphic behaviour. Even though not specified in Recommendation X.720, it is necessary that the naming structure is the same for the two classes. Even if the managed object class parameter refers to an allomorphic class, in order for the agent to recognize the managed object, it is necessary that the managed object field uses the same structure for the actual class and the allomorphic class. The same structure implies that the sequence for constructing the local and distinguished names are the same (the superior class and RDN attributes are the same for the extended and allomorphic classes). This condition is satisfied in most cases when the two classes are related by inheritance (the name binding can include the phrase AND SUBCLASSES).

The example of the multi-port circuit pack is one where the structure rules for naming is the same as the circuit pack even though the former could not be derived as a subclass (because of the extension in permitted values). Thus, from naming perspective, it is possible to consider an instance of multi-port circuit pack to be allomorphic to circuit pack object class.

I.5 Examples of the use of allomorphism

This clause contains examples of the scenarios where the managing and managed systems implement to different versions of an information model. As a flash cut of all systems to the same version is not possible, using allomorphism to support interoperability between the systems will become important.

The figures assume that agent and manager systems are from multiple suppliers. Thus, the release numbers and the relationship to implemented version of the information model are not correlated among the suppliers.

Figure I.2 describes the simple scenario. The schema for management information model (SMK) corresponds to exactly the same definitions (both are from the interface perspective at the same release). Different numbering is used to convey the possibility that when different vendors supply the systems, they may use different release numbering options.

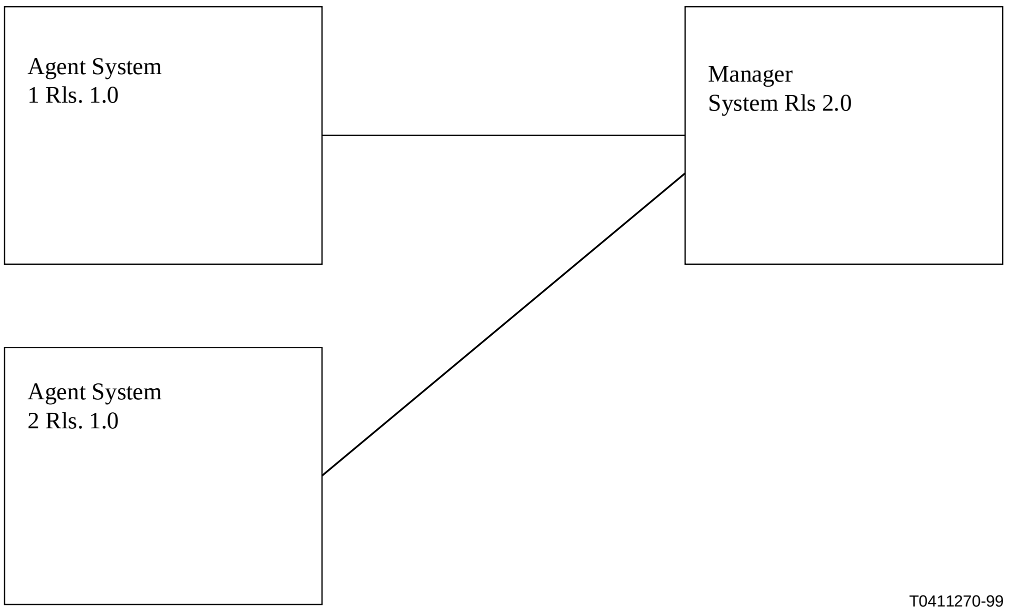

Figure I.2/Q.812 – Scenario 1

In this case, interoperability does not require the support or otherwise of allomorphism. Both agent systems and manager system are not expected to send or receive management information different from that defined by the schema.

Figure I.3 describes the following scenario: The schema for the management information model (SMK) the manager implements has more capability than Agent System Rls 1.0. The management system manages more than one agent system (different suppliers). Agent System 2 Rls 1.5 and Manager System Rls 3.0 implements the same features (SMK is the same from interface perspective). Agent System 2 was also managed by Manager System 2 which did not upgrade to include the new features.

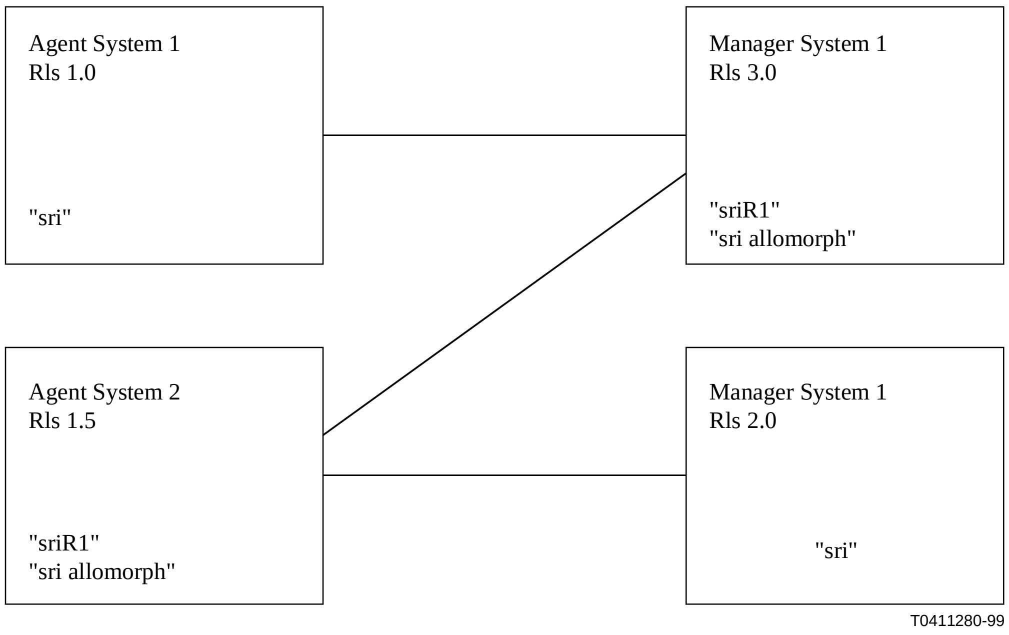

Figure I.3/Q.812 – Scenario 2

Case 1: Manager Provided Interoperability: For simplicity let us take a single managed object class "sri" and an extended class "sriR1. The interoperability solutions can be explained using this simple case without loss of generality (even though a system may choose to support allomorphism for some classes and not others). Assume that Manager System in Rls 3 can manage extensions that are not offered by Agent System 1 but by Agent System 2.

Case 2: Support or otherwise of allomorphism by Agent System 1 is not relevant. Agent System 2 supports allomorphism and has implemented the class "sriR1". The allomorph attribute contains the value "sri". Interactions between Manager System 1 and the two agents are the same as in case 1. Manager 2 does not understand the additional capabilities in "sriR1" and for notifications from Agent System 2 (using the extended class sriR1), the behaviour may not be the same as in case 1. For notifications only defined for sriR1, Manager 2 will not know how to process them and should therefore ignore them as far as management activity is concerned. Because Agent System 2 supports allomorphism, when the manager requests an operation using class "sri", it will perform the operation according to specifications for the actual class "sriR1". The agent may inform the manager that the actual class is "sriR1" by including it in the response (this is not required if the request was directed to a specific instance). Based on the requested operation, the information provided will match that of the actual class. Manager 2 should be able to ignore unrecognized information without disrupting the association. This implies that the manager should provide for some minimum level of interoperability.