3GPP TR 32.830 V2.0.0 (2012-12)

Technical Report

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Study on version handling over Itf-N (Release 12)

Keywords
NRM, IRP, Converged Management,Version Handling

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

Introduction

A number of issues with version handling in the current set of SA5 OAM specifications have recently been detected. It is therefore important to make an inventory of the current situation, identify issues and recommend a solution.

1 Scope

The scope of the study is to:

- Document current version handling capabilities in SA5 OAM specifications (clause 5)
- Identify and agree on the use cases and requirements (clause 6) for a coherent approach for version handling capabilities for SA5 OAM specifications
- Identify solution(s) with pros and cons to support the identified requirements (clause 7). The solutions may require enhancement or modification of rules in the IRP methodology as well as enhancements of existing or addition of new IRP specifications, utilizing SOA capabilities where appropriate. This includes:
    1. The identification of entities that require version control (clause 7.1)
    2. The protocol(s) by which the entity version can be discovered and registered (clause 7.2)
- Recommend a solution (clause 8).

Note: "Version handling" in the context of this study addresses what kind of support for version information that is available over the Itf-N interface, i.e. what the IRPManager can "see" regarding which versions of the 3GPP specifications that are supported by the IRPAgent, for which object instances etc.

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 TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 32.150: "Telecommunication management; Integration Reference Point (IRP) Concept and definitions".

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

[4] 3GPP TS 32.362: "Telecommunication management; Entry Point (EP) Integration Reference Point (IRP); Information Service (IS)".

[5] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[6] 3GPP TS 32.662: "Telecommunication management; Configuration Management (CM); Kernel CM Information Service (IS)".

[7] 3GPP TS 32.602: "Telecommunication management; Configuration Management (CM); Basic CM Integration Reference Point (IRP): Information Service (IS)".

[8] 3GPP TS 32.612: "Telecommunication management; Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Information Service (IS)".

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

[10] 3GPP TS 32.342: "Telecommunication management; File Transfer (FT) Integration Reference Point (IRP); Information Service (IS)".

[11] 3GPP TS 32.423 Subscriber and equipment trace; Trace data definition and management.

[12] 3GPP TS 32.311: "Telecommunication management; Generic Integration Reference Point (IRP) management; Requirements".

[13] 3GPP TS 32.111-2: "Telecommunication management; "Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

[14] 3GPP TS 32.102: "Telecommunication management; Architecture".

[15] 3GPP TS 32.300: "Name convention for Managed Objects"

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, please refer to 3GPP TS 32.101 [3], 3GPP TS 32.102 [14] and 3GPP TS 32.150 [2].

3.2 Abbreviations

For the purposes of the present document, please refer to 3GPP TS 32.101 [3], 3GPP TS 32.102 [14] and 3GPP TS 32.150 [2].

4 Concepts and background

The majority of SA5 specifications today are specified as Integration Reference Points (IRPs) following the concept and methodology outlined in the 32.15x-series specifications. There are three categories of IRP specifications (see formal and more detailed definitions in clause 3.1 of 3GPP TS 32.150 [2]):

- Interface IRPs
- NRM IRPs
- Data Definition IRPs

5 Current version handling capabilities

5.1 General

The version handling for Interface IRPs and NRM IRPs is managed with different operations and parameters. Version handling for the Data Definition IRPs does not exist.

However, there are a few specifications that are not IRPs and which do not follow the same version handling concept. Some specifications do not have any version handling support at all e.g.: Performance measurements, KPI and Subscriber and Equipment Trace specifications.

Furthermore, the SOA architectural concept has been introduced in the 3GPP Management Reference Model – see clause 5.4 of 3GPP TS 32.101 [3]. There is a need to consider if there are any additional needs for version handling support with respect to SOA.

5.2 Interface IRPs

5.2.1 Generic IRP Management

3GPP TS 32.311[12] (Generic IRP Management): Generic parameter definition "**IRP document version number string**" (a.k.a "IRPVersion").

3GPP TS 32.311 [12] and 3GPP TS 32.312 [5]: Generic version retrieval function **`GetIRPVersion()`.** This operation is inherited by all "domain specific" Interface IRPs, such as Alarm IRP IS [13], and mapped to local specific operation/method names in the respective Solution Sets. This operation is used for the IRPManager to find out which **Interface** IRP SS versions that are supported by an IRPAgent.

The definition of `GetIRPVersion` is as follows: "*IRPManager wishes to find out the IRP SS versions supported by an IRP. The IRP shall respond with a set of supported IRP SS version(s). The list of returned IRP versions is such that the IRPManager can use any of these versions without having to specify an IRPVersion to the IRPAgent*.".

The definition of the operation's output parameter (except the "status") named `versionNumberSet` is: "*It indicates one or more SS version numbers (`IRPVersion`, as defined by "IRP document version number string" in clause 3.1) supported by the IRP.*"

5.2.2 Entry Point IRP (EP IRP)

EP IRP in 3GPP TS 32.362 [4] has an operation named `getIRPOutline`. It takes as input an IRPVersion parameter which specifies one or all supported IRPVersions. As output is returned a `supportedIRPList` parameter.

`getIRPOutline` is defined as follows:

The IRPManager uses this operation to request the EPIRP to return the outline information of the supported IRPs. IRPManager could set a filter constraint on the returned information according to specific requirements.

The EPIRP shall return the outline information of all the IRPs, including itself and other EPIRP instances that it knows, supported by the IRPAgent that contains the EPIRP.

The EPIRP may additionally return the outline information of all the IRPs, including EPIRP instances, supported by other IRPAgents.

5.3 NRM IRPs

5.3.1 Operations for NRM IRP versions

3GPP TS 32.662 (Kernel CM IRP): Operation **getNRMIRPVersion.** This operation is used for the IRPManager to find out which **NRM** IRP SS versions that are supported by an IRPAgent. However, it does not indicate which NRM versions that are supported in each of the instantiated managed objects of the MIB.

The first and main part of the definition of `getNRMIRPVersion` reads as follows:

"When the IRPManager invokes `getNRMIRPVersion` to find out the Network Resource IRP SS document versions (IRPVersions) supported by the IRPAgent, the IRPAgent shall respond, via the `versionNumberList` output parameter, with a list of supported Network Resource IRPversions.
The rule for deriving the IRP document version number string is defined in 3GPP TS 32.311 [12].
An example of this return value can contain two IRPVersions, where one indicates the 3GPP Generic Network Resource IRPVersion (e.g. "32.623 V4.2" in case of CORBA implementation) while the other indicates the 3GPP UTRAN Network Resource IRPVersion (e.g. "32.643 V4.1" in case of CORBA implementation).

It is expected that vendors may provide vendor-specific extended capabilities and features (VSE) that are based on a 3GPP published specification. It is further expected that the vendor will publish these VSE in a document with an unambiguous identification. ".

`getNRMIRPVersion` returns two parameters (except the "status"):

- `versionNumberList`: "It carries one or more SS version numbers supported by this IRP agent.", and
- `vSEVersionNumberList`: "It carries one or more identifications of vendor published documents containing VSE NRMs specifications.".

5.3.2 Generic NRM IRP

The two IOCs ManagedElement and ManagementNode contain the following attribute: **swVersion**. It is defined as "The software version of the `ManagementNode` or `ManagedElement` (this is used for determining which version of the vendor specific information is valid for the `ManagementNode` or `ManagedElement`).".

The IOC `VsDataContainer` contains the following attribute: **vsDataFormatVersion.** It is defined as "Name of the data format file, including version.".

5.4 Data Definition IRPs

5.4.1 State Management IRP

Version handling for the Data Definition IRPs does not exist (currently State Management IRP is the only DD IRP).

5.5 Other management definitions

5.5.1 Performance measurements and KPIs

PM file format has version handling, consisting of

- File format version
- NE SW version

Measurement definitions do not have version handling.

KPIs are FFS.

5.5.2 Trace functions

The Trace file XML schema (see 32.423 [11] Annex A.1) file header contains a version number.

5.5.3 SOA entities

FFS.

5.6 Release 9 features

This clause gives an overview of the current version handling capabilities by listing the 3GPP Release 9 features/capabilities allowing IRPManager to discover version information of certain managed entities, e.g. PM file format version, IRPVersion of an YyyIRP, etc.

Features	Target entity to which the IRPManager sends the discovery request	References
Discover IRPVersions supported,	An Interface `YyyIRP` instance (e.g. `AlarmIRP`)	See `getIRPVersion`(), 6.3.1 of [5]. Note that all Interface `YyyIRP` inherits from `ManagedGenericIRP`, 5.3.1 of [5], that defines `getIRPVersion`(…).
Discover the outline information of the known (by the target entity) supported IRPs, including `IRPVersion` information.	`EPIRP`	See `getIRPOutline(iRPVersion…)`, 6.3.1 of [4].
Discover supported IRPVersions and supported vendor-specific versions that are related to NRM IRP.	`KernelCMIRP`	See `getNRMIRPVersion()`, 7.3.1 of [6]).
Discover SW version that is used for the specified IOCs.	`BasicCMIRP`, `BulkCMIRP`	See `getMoAttributes(swVersion…)`, 7.3.1 of [7]. Note that only `ManagedElement` IOC and `ManagementNode` IOC contains attribute `swVersion()`. See 6.1.3.3 and 6.1.3.5 of [9].
Discover version information related to vendor-specific capabilities.	`BasicCMIRP`, `BulkCMIRP`	See `getMoAttributes(vsDataFormatVersion…)`, 7.3.1 of [7]. Note this is applicable when the IOC name-contains a `VsDataContainer`. The `VsDataContainer` IOC has the attribute `vsDataFormatVersion`. See 6.1.3.9 of [9].
Discover the File Format version of a particular file when file is created.	`FTIRP`	Receive and process the `NotifyFileReady`, 6.5.1 of [10]. It carries `fileInfoList` that contains the File Format information.
Discover the File Format version of files ready for retrieval.	`FTIRP`	See `listAvailableFiles`, 6.3.1 of [10]. Its output parameter `fileInfoList` contains the File Format information.

5.7 Issues

The following issues and inconsistencies with version handling have been identified in the current set of SA5 specifications.

Issue 1: Network Resource Model (NRM) object version handling

In today's set of standard IRP specifications, there is support for an IRPManager to retrieve a list from the IRPAgent about which IRPVersion(s) (of the NRM IRP SSs) that the IRPAgent supports, one or more. But if the IRPAgent supports more than one IRPVersion, there is no standardised way to know which IRPVersion that a particular Managed Object (MO) instance belongs to. Thus, there is an information gap on Itf-N which needs to be filled.

Issue 2: Version handling of Interface IRPs versus NRM IRPs

For IRPManager to obtain the IRPVersion(s) of an IRPAgent's supported Interface IRPs and supported NRM IRPs, IRPManager needs different operations. It might be beneficial for IRPManager to use identical/similar/same operation (to achieve some level of consistency) to obtain the two different kind of information.

Issue 3: Version handling of management information such as alarms, measurements and trace data

Currently there is no version handling defined in SA5 for management information such as alarms, measurements and trace data.

Issue 4: SOA support

As SA5 have recently (Rel-9) introduced SOA (Service Oriented Architecture) for IRPs, SA5 should also study if and how the version handling can satisfy the needs of SOA, and how SOA may provide capabilities for a coherent version handling (e.g. registration & discovery).

6 Use cases and requirements

The objective of this clause is to Identify and agree on the use cases and requirements for a coherent approach for version handling capabilities for SA5 OAM specifications.

6.1 Use cases

6.1.1 Overview

Questions to ask & investigate: What 'things' need version control? What are the UCs that can illustrate the reason why IRPManager needs to know the versions of the 'things' in run-time?

This section lists the potentially useful Use Cases for version information. See Appendix A for their detail sequence diagrams.

1. IRPManager requests a list of all supported interface IRPs and versions
2. IRPManager requests the IRP reference (i.e. address of the Interface IRP such as AlarmIRP).
3. IRPManager requests list of supported Interface IRP IRPVersions.
    - 3a Using the EPIRP
    - 3b Direct to Interface IRP (e.g. Alarm IRP)
4. IRPManager requests a list of supported Interface IRP (e.g. AlarmIRP) operations (a. supported mandatory operations b. supported optional operations)
5. IRPManager requests a list of supported Interface IRP (e.g. AlarmIRP) notifications (a. supported mandatory operations b. supported optional operations)
6. IRPManager requests a list of supported Interface IRP (e.g. AlarmIRP) packages
7. IRPManager requests a list of supported parameters for operations of Interface IRP (e.g. AlarmIRP)
8. IRPManager requests a list of supported parameters for notifications of Interface IRP (e.g. AlarmIRP)
9. IRPManager requests a list of all supported NRM IRP IRPVersions
10. IRP Manager requests a list of all supported UTRAN NRM versions (special case of UC9)
11. IRP Manager requests a list of all supported UTRAN NRM IOCs
12. IRPManager requests a list of all supported UtranCell (UTRAN NRM) attributes (M/O)
13. IRPManager requests a list with IRPVersion of each NRM object instance. See more detailed description below.
14. IRPManager requests the IRPVersion of the supported PM file format, i.e. the XML file format definition and the ASN.1 file format definition.
15. IRPManager request the IRPVersion of supported PM measurement types.
16. Version handling of Trace related information: FFS
17. Version handling of SOA related information: FFS

6.1.2 More detailed description and justification of use cases

6.1.2.1 Use Case 13 (NRM object instance versions)

This use case addresses Issue 1 in subclause 5.7 (also in the agreed WID for this study):

Issue 1: Network Resource Model (NRM) object version handling

> In today's set of standard IRP specifications, there is support for an IRPManager to retrieve a list from the IRPAgent about which IRPVersion(s) (of the NRM IRP SSs) that the IRPAgent supports, one or more. But if the IRPAgent supports more than one IRPVersion, there is no standardised way to know which IRPVersion that a particular Managed Object (MO) instance belongs to. Thus, there is an information gap on Itf-N which needs to be filled.

Below follows an analysis of the network deployment/update scenarios causing this issue:

In the course of interactions between IRPManager and IRPAgent, there are moments (e.g. when reading/writing instance attributes) and situations where the IRPManager needs to know the class definition(s) of IOC instance(s).

Examples of the situations are:

Example 1: When some but not all instances of an IOC definition have been upgraded (a normal occurrence in a large and evolving managed network, as it is often impossible to upgrade all network nodes with a new SW version at the same occasion). In more detail it may look like this (split into two cases, 1A and 1B):

1A: The IRPManager and IRPAgent are designed to manage only one version of the NRM.

a. Operator has an LTE network covering 15 provinces. Each province has one Network Manager (NM) including an IRPManager, and 5-10 Element Managers (EMs). Each of the EMs has an IRPAgent using the E-UTRAN NRM IRP (SS definitions in 3GPP TS 32.766). All IRPAgents in the country use NRM SS version 10.5.0, and the complete network including EMs is supplied by one vendor.

b. The vendor delivers a new SW version for a) the eNodeB and b) the EM supporting version 10.7.0 of 32.766. There are several differences between the IOC definitions in version 10.5.0 and 10.7.0, e.g. two attributes representing two new radio parameters have been added to EUtranGenericCell. Corresponding modifications to support the two new radio parameters have been included in the new eNodeB SW package.

c. It is up to the operator when to deploy/upgrade the new SW in all eNodeBs and EMs throughout the network. The operator chooses to upgrade one province at a time. Since the IRPAgent can only support one version, the upgrade in each province will have to be made for the EM and all eNodeBs during one night. Thus, during the 15-day SW upgrade period some provinces will have version 10.5.0 and some will have 10.7.0 of the E-UTRAN NRM IRP (and corresponding network installation).

d. The operator has also in parallel developed a new NMS SW with IRPManager support for version 10.7.0 of 32.766. The new IRPManager can only support and understand version 10.7.0, so the operator has two choices: (Upd1) Install it before the network upgrade starts or (Upd2) install it after all EMs are upgraded. If Upd1 is chosen, when this IRPManager attempts to access e.g. an EUtranGenericCell instance of version 10.5.0 it will cause a 'failed access attempt' or "type mismatch" alarm, every time. Vice versa, if Upd2 is chosen, when this IRPManager attempts to access e.g. an EUtranGenericCell instance of version 10.7.0 it will cause a 'failed access attempt' or "type mismatch" alarm, every time. Thus, during the 15-day SW upgrade period, access attempts by the IRPManager to some provinces will work, while for some provinces it will cause lots of alarms as well as inability to update or even read the attribute values of some MOs.

1B. The IRPManager and IRPAgent are designed to manage two parallel versions of the NRM

a. Operator has an LTE network covering 15 provinces. Each province has one Network Manager (NM) including an IRPManager, and 5-10 Element Managers (EMs). Each of the EMs has an IRPAgent using the E-UTRAN NRM IRP (SS definitions in 3GPP TS 32.766). All IRPAgents in the country use NRM SS version 10.5.0, and the complete network including EMs is supplied by one vendor.

b. The vendor delivers a new SW version for a) the eNodeB and b) the EM supporting version 10.7.0 of 32.766. There are several differences between the IOC definitions in version 10.5.0 and 10.7.0, e.g. two attributes representing two new radio parameters have been added to EUtranGenericCell. Corresponding modifications to support the two new radio parameters have been included in the new eNodeB SW package. The IRPAgent in the upgraded EM can now support both version 10.5.0 and 10.7.0 of the NRM, i.e. it can manage a MIB where some instances of eNodeBFunction implement NRM version 10.5.0 and some instances implement version 10.7.0.

c. It is up to the operator when to deploy/upgrade the new SW in all eNodeBs and EMs throughout the network. The operator chooses to upgrade each province over a period of three nights. Thus, during the 45-day SW upgrade period some provinces will have version 10.5.0 and some will have 10.7.0 of the E-UTRAN NRM IRP (and corresponding network installation), and even within each province there will be a mix of both versions. But this is no problem for the IRPAgent which can handle both versions.

d. The operator has also in parallel developed a new NMS SW with IRPManager support for version 10.7.0 of 32.766 and installed it before the network upgrade starts. It can support and understand both version 10.5.0 and 10.7.0. But, there is no (standardised) information showing which IRPAgent or which instance of eNodeBFunction that implements which NRM version. However, this IRPManager must know in advance, for every accessed NRM IOC instance, which version of the class the instance supports. Because the IRPManager is designed with two processes - process P1 supporting version 10.5.0 and process P2 supporting 10.7.0. And if P1 attempts to access an EUtranGenericCell instance of version 10.7.0, it will cause a 'failed access attempt' or "type mismatch" alarm, every time. Similarly, if P2 attempts to access an EUtranGenericCell instance of version 10.5.0, it will cause a 'failed access attempt' or "type mismatch" alarm. Thus, during the 45-day SW upgrade period, access attempts by the IRPManager to some eNodeBs will work, while for some other eNodeBs it will cause lots of alarms, since the IRPManager has no way of knowing in advance which instance of eNodeBFunction that implements which NRM version. The only way to know it is by "trial and error" which will cause a huge amount of failed attempts and alarms.

Example 2: When a managed network contains some instances whose IOC definitions are vendor-specific. This is the case when the vendor-specific attribute(s) of the IOC, e.g. `MscFunction`, is not captured in `VsDataContainer` [1] (that is name-contained by `MscFunction`) but captured in `MscFunction` whose definition is defined by the vendor itself, and not by 3GPP. In more detail it may look like this:

<Same scenarios as for Example 1 Case 1A and 1B above, with upgrade of the vendor-specific definition of MscFunction from an older version to a newer version >

6.1.3 Current 3GPP SA5 TS support for the use cases

The following Table is a summary of the current TS support level of the above UCs.

Legend: NS = Not Supported; NA = Not Applicable

Table 6.1-1: Summary of the current TS support level of above Use Cases

	EPIRP	Interface IRP (e.g. AlarmIRP)	KernelCMIRP	Bulk/Basic CM IRP
UC1 (IRP's & Versions)	Supported	NA	NA	NA
UC2 (IRP Reference)	Supported	NA	NA	NA
UC3a (IRP Version)	Supported	NA	NA	NA
UC3b (IRP Version)	NA	Supported	NA	NA
UC4 (supported operations)	Supported	Supported	Supported	Supported
UC5 (supported notifications)	Supported	Supported	Supported	Supported
UC6 (IRP Packages)	NA	NA	NA	NA
UC7 (parameters of an operation)	Supported	Supported	Supported	Supported
UC8 (parameters of a notification)	Supported	Supported	Supported	Supported
UC9 (NRM's and versions)	NS	NA	Supported	NA
UC10 (one NRM and versions)	NS	NA	Partially Supported	NA
UC11 (NRM IOCs)	NS	NA	NS	NS
UC12 (NRM object attributes)	NS	NA	NS	NS
UC13 (NRM object instance versions)	NS	NA	NS	NS
UC14 (PM file format)	NA	NS	NA	NA
UC15 (measurement types)	NS	NS	NA	NA
UC16 (Trace info)	NA	Supported (file version)	NA	NA
UC17 (SOA info)	FFS	FFS	FFS	FFS

6.2 Requirements

1. Identify one or more alternative solutions for all the non-supported Use Cases in the above table, except the ones marked Non-practical.

2. Identify one or more alternative solutions, if possible, to define a common function (operation) for all the supported version handling capabilities, with their pros and cons.

7 Solution(s)

The objective of this clause is to identify solution(s) with pros and cons to support the identified requirements. The solutions may require enhancement or modification of rules in the IRP methodology as well as enhancements of existing or addition of new IRP specifications, utilizing SOA capabilities where appropriate.

7.1 Entities that require version control

Entities that require version control: As defined in subclause 6.2.

7.2 Solution alternatives for currently non-supported Use Cases in clause 6.1

7.2.1 UC11 (NRM IOCs)

TBD

7.2.2 UC12 (NRM object attributes)

TBD

7.2.3 UC13 (NRM object instance version)

7.2.3.1 Introduction

3GPP TS 32.300 [15] is about name spaces. The name space specified in [15] is the "container" of identifiers of class instances.

Quoted from http://en.wikipedia.org/wiki/Namespace, "…a **namespace** is a container that provides context for the identifiers (names, or technical terms, or words) it holds, and allows the disambiguation of homonym identifiers residing in different namespaces".

In this document, the "name space" is not about the "container" of identifiers of class instances. Rather, it refers to the "container" of identifiers of classes.

A number of solution proposal options are presented here.

7.2.3.2 Option 1

In this solution, unique class name space names are assigned to organizations (e.g., standard bodies, vendors) by the Internet Corporation for Assigned Names and Numbers (ICANN), a non-profit corporation for assigning and managing, among other things, name spaces. The assigned name space can be further extended (i.e. subdivided) by the organization to create its own hierarchical name spaces.

This solution does not require modification of the DN structure. This solution requires:

- A new attribute, called nsVersionInfo, is defined in Top IOC (which is inherited by all other IOCs).
- Also to consider (for discussion): Having the new nsVersionInfo attribute in Top IOC means that all instances (could be hundreds of thousands of them in a large installation) will have it. That would mean in a large installation, lots of attributes with the same value data. We should consider a better way of implementing this attribute (on IS and/or SS level) – see Options 2a, 2b and 3 below.

If and when an IRPManager wants to be certain of the class namespace and version of an instance, it reads this new attribute (assuming IRPAgent implements this attribute). The attribute value could be "RED1.0", "WHITE2.1" etc., explicitly indicating the class namespace and version.

The legal values (and their semantics) of "RED1.0", "WHITE2.1" etc. need clarification. For example, should the standard define legal values for all, some or none of these values?

The "RED1.0", "WHITE2.1" etc. are fictitious names for a string representing a specification authority, a specification number and a version number. Examples of legal values could be:

- "3GPP, 32.766 V9.1" where the first component is the name of the specification authority and the 2nd component is the IRPVersion string (as defined in [3]) of the specification where the subject class is defined.
- "MEF, 7.1" that identifies the "Technical Specification MEF 7.1, Phase 2 EMS-NMS Information Model" published by MEF (the name of the specification authority).
- "www.acme.com, turbo 3.4.112" that identifies the specification named "turbo 3.4.112" published by an organization identified by "www.acme.com".

7.2.3.3 Option 2a

This option builds on option 1 regarding the namespace and version definition format and semantics, but with the following differences removing the redundancy of the version information.

Detailed description:

1. A new IOC is defined, e.g. named NSV. It is optionally name-contained by SubNetwork and ManagedElement (with an {xor} constraint between these two containment relations), and contains a **multi-valued** attribute named NsVersionInfo holding the default namespace and version information of all supported NRM SSs. The contents and legal values of each element of NsVersionInfo is according to the description in Option 1 above.

2. The root instance of each MIB portion (SubNetwork or ManagedElement) directly name-contains one NSV instance. This "root NSV instance" identifies the list of default namespace and version values for all NRM IRP SSs supported by the MOs in the MIB portion contained by this root SubNetwork or ManagedElement instance.

3. Optionally, other NSV instances may be contained by the "root NSV instance". These NSV instances are all directly name-contained the "root NSV instance", and are used to identify fragments of the MIB which deviate from the default namespace and version values.

4. For each fragment (sub-tree) of the MIB where any deviation from the default namespace and specification versions exists, a new instance of NSV is created with an association directly to the root (MO) of this MIB fragment, which can be any MO in the MIB. The NsVersionInfo of that NSV instance contains the namespace and version of the NRM SSs deviating from the default value(s) (given by the "root" NSV instance), and supported by the MOs of this fragment. Deviations associated with fragments on "lower containment level" will over-ride deviations identified for fragments higher up in the same branch of the containment tree.

Below is an example illustrating the use of the NSV instances. Figure 1 is a fictitious MO tree.

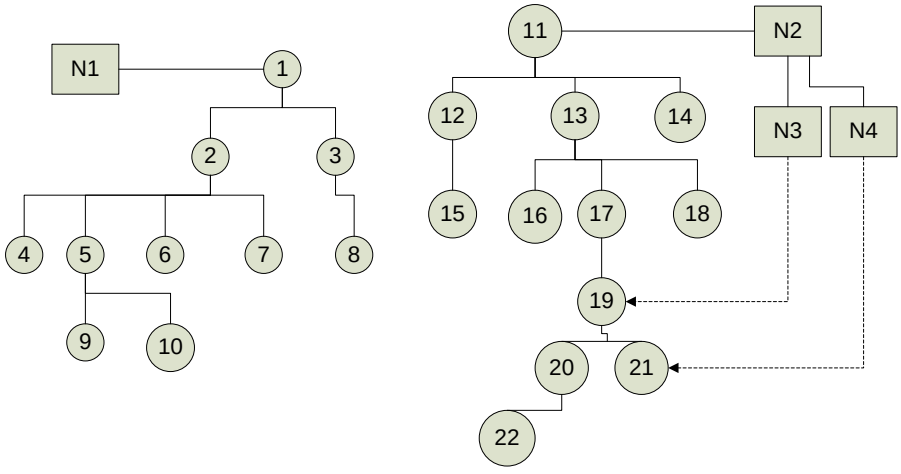


Figure 1: Example containment tree of MIB instances

Example 1:

NSV instance N1 (contained by MO 1) contains NsVersionInfo attribute with the list of two namespace-versions (a,b). This means that class definitions for instances 1..10 should be found in namespace-versions a and b.

Example 2:

NSV instance N2 (contained by MO 11) contains NsVersionInfo attribute with the list of namespace-versions (a,b,c). NSV instance N3 (contained by N2 and with an association to MO 19) contains NsVersionInfo attribute with the list of namespace-versions (d,e). NSV instance N4 (contained by N2 and with an association to MO 21) contains NsVersionInfo attribute with the single element of namespace-versions (a).

This means that

- class definitions for instances 11..18 should be found in namespace-versions a, b and c,
- class definitions for instances 19, 20 and 22 should be found in namespace-versions d and e, and
- class definition for instance 21 should be found in namespace-version a.

7.2.3.4 Option 2b

This option also builds on option 2a regarding the namespace and version definition format and semantics in the NsVersionInfo attribute, as well as identifying a) default namespace and version information of all supported NRM SSs and b) deviating namespace and version information in the fragments of the MIB, if any, where it exists.

The main difference compared to option 2a is that in 2b, the NsVersionInfo attribute is defined in one of the Inventory NRM IRP IOCs, e.g. InventoryUnit, thereby reusing the structure of the Inventory NRM which the IRPManager already may be using to identify "administrative" type of information about the MOs which is semi-permanent (not updated frequently), and mirroring a large portion of the MOs even if not all.

One issue with this option is that InventoryUnit (and other IOCs in this NRM) can only point to ManagedFunction and its derivatives. Therefore this option cannot identify "versions deviating from the default" for those MOs which are not derived from ManagedFunction, and there are a few, e.g.: UtranRelation, EP_RP, EP_IuCS, EP_IuPS, EP_Iur, CellOutageCompensationInformation, EUtranCellNMCentralizedSON. Possible ways to get around this issue could be to:

a) Assume that all MOs whose MOCs are not derived from ManagedFunction are of the same version as the closest (or topmost) MOC in the same containment branch defined in the same NRM SS.

b) Define a <<ProxyClass>> ManagedEntity in the Inventory NRM IRP to which InventoryUnit has an association, representing an optional association to any IOC in any NRM IRP.

Another issue with this option is that InventoryUnit (and other IOCs in this NRM) are all contained by ManagedElement, so definition of default namespace and version info in a "root InventoryUnit instance" would be valid per ManagedElement instance. So it must be defined once per ManagedElement instance, which can be a large number, and cannot be set up once per SubNetwork instance. Further, it cannot define the version information for MOs name-contained by SubNetwork instead of ManagedElement, such as ExternalENBFunction and ExternalEUtranGenericCell. Possible ways to get around this issue could be to:

c) Define a new IOC contained by SubNetwork and holding the NsVersionInfo attribute, in the Inventory NRM. But that would mean practically the same solution as option 2a.

d) Let the InventoryUnit(s) in one and only one of the ManagedElement instances define the default and possibly deviating versions for all MOs contained by SubNetwork and not ManagedElement.

e) "Move" the containment of InventoryUnit up to be directly name-contained by SubNetwork.

A third issue with this option is that it is less straightforward to identify which instance of InventoryUnit that contains the default namespace-version values, since InventoryUnit does not have an association to ManagedElement and there could be 0..n instances of InventoryUnit directly name-contained by one ManagedElement. Possible ways to get around this issue could be to:

f) Add an association from InventoryUnit to ManagedElement IOC (or to SubNetwork, if option e) above is chosen).

g) Limit the cardinality for instances of InventoryUnit directly name-contained by one ManagedElement, to 0..1.

Detailed proposal for option 2b:

1. Add an NsVersionInfo attribute to the InventoryUnit IOC.
2. Choose one of the options a) and b) above, one of the options c), d) or e) above, and one of the options f) and g) above.

7.2.3.5 Option 3

This option uses a new operation called getIRPVersionMap. This operation can be defined as a new operation for Kernel CM IRP.

The NM invokes this `getIRPVersionMap` operation to discover the IRPVersion of the IOC definition used for one, some or all MOs in the MIB.

This operation does not require any input parameter.

The IRPAgent shall respond, via the `map` output parameter carrying the requested information. The `map` parameter is defined as a list of elements where each element is an ordered list of (DN, pIdentifiers, scope) defined in Table 7.2-1 below.

An MO (i.e. the subject) in the MIB has its DN.

If there is one element (of the ordered list) that carries the subject's DN, then the pIdentifier of that element would be applicable to the subject.

If there is no element that carries the subject's DN but there is one or more elements that a) carry the DN of the subject's superiors (directly or indirectly) and b) carry a scope that is FALSE, then pick the element that identifies an MO closest to the subject in the name-tree. The pIdentifier of the picked element would be applicable to the subject.

Table 7.2-1: Output parameters

Parameter	Description
dN	This carries the Distinguished Name of the MO. In a map, a particular DN cannot occur more than once.
pIdentifiers	This carries one or more IRPVersion (namespace-version). Each identifies a specific specification, of specific publication date, that contains the IOC definition.
scope	This holds either true or false. The value true indicates that the IOC definitions, specified in the specifications identified, should only apply to the MO identified by the dN above. The value false indicates that the IOC definitions, specified in the specifications identified, should apply to the MOs that form a tree whose top tree-node is the MO identified by the dN above (in Figure 3 as an example, instance-5 is the top tree-node of a tree that is made up of instance-5, 9 and 10).

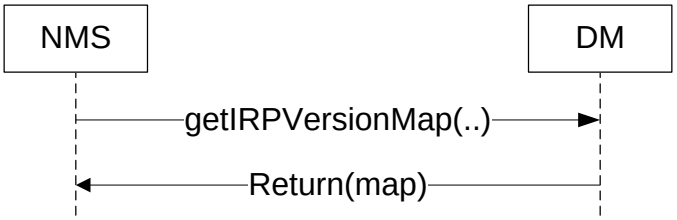


Figure 1: Sequence diagram for `getIRPVersionMap`

Below are some examples illustrating the use of the map information. Figure 3 is a fictitious MO tree.

Example 1: map = (1,(a,b),FALSE)

Class definitions for instances 1..10 should be found in namespace-versions a,b. Class definitions for instances 11..22 are not known.

Example 2: map = (1,(a,b),FALSE; 5,k,TRUE)

Class definitions for instances 1..10 except 5 should be found in namespace-versions a,b. Class definition for instance 5 should be found in namespace-version k and if not, should be found in namespace-versions a,b. Class definitions for instances 11..22 are not known.

Example 3: map = (1,(a,b),FALSE; 11, (a,b,c), FALSE, 13,d,TRUE, 17,e,FALSE))

Class definitions for instances 1..10 should be found in namespace-versions a,b.

Class definitions for instance 11..22, except instances 13, 19..22, should be found in namespace-versions a,b,c. Class definition for instance-13 should be found in namespace-version d; if not, should be found in namespace-versions a,b,c. Class definition for instances 19..22 should be found in namespace-version 3; if not, should be found in namespace-versions a,b,c.

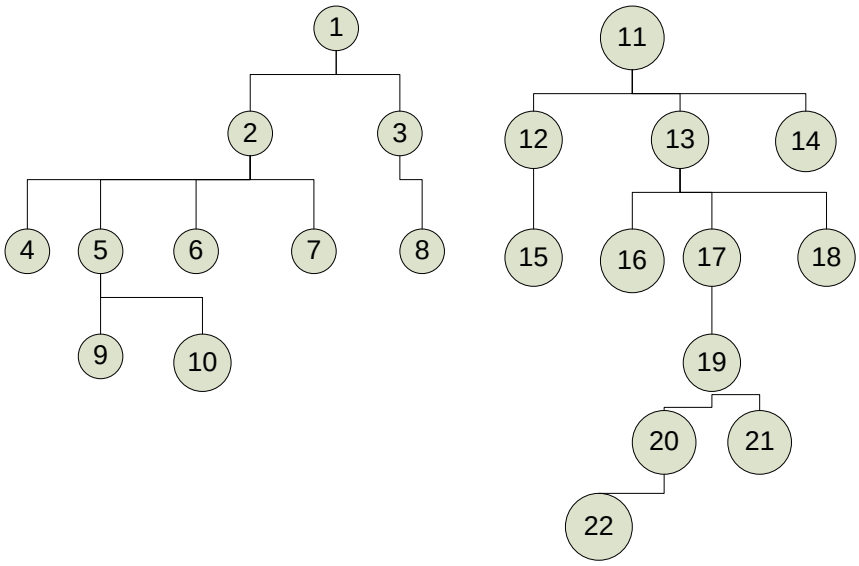


Figure 3: Example containment tree of MIB instances

A variation of getIRPVersionMap

The `getIRPVersionMap` described above requires no input parameter. Here is a variation of `getIRPVersionMap` that requires an input parameter.

The usage of `getIRPVersionMap` without input parameter is to allow NMS to indicate that it wants information on all MOs. The usage of this variation of `getIRPVersionMap` with parameter is to allow NMS to indicate that it wants information on some MOs (e.g. instance-8 and instance-16 of Fig. 3) only or on some MOs of branches of the name tree only (e.g. instances-5,9,10 of Fig. 3).

The input parameter is a list of elements where each element is a set of (DN, scope) as defined below:

Table 7.2-2: Elements of the input parameter

Parameter	Description
dN	This carries the Distinguished Name of the MO. The input parameter cannot have the same DN appearing more than once.
scope	This holds either true or false. The value true indicates that NMS requests the namespace-version for an MO identified by the dN above. The value false indicates that NMS requests the namespace-version(s) for MOs that form a tree whose top tree-node is the MO identified by the dN above (in Figure 2 as an example, instance-5 is the top tree-node of a tree that is made up of instance-5, 9 and 10).

The output parameter is a map as defined above.

7.2.3.6 Comparison (pros/cons) of the options

Pros/cons for Option 1:

Pros:

- Simple solution that completely reuses the existing NRM structure.

Cons:

- Will imply that the new attribute NsVersionInfo will be defined in every MO in the MIB, which can be a huge number, and it generates a lot of redundancy since most of the instances will have the same value for this attribute.
- Not backward compatible as all existing IOCs/MOCs will be modified.
- Does not work with the (currently existing) limitation that Basic CM IRP does not support reading of (one or more) selected attributes of an MO instance in one operation (the only supported semantics is “read all attributes in one shot”).

Pros/cons for Option 2a:

Pros:

- Reduces the data amount significantly. That also means reduced load on the IRPManager, IRPAgent and network in access operations over Itf-N.
- Is backward compatible in that no existing IOCs (including Inventory NRM) are modified.

- Works independently of the (currently existing) limitation that Basic CM IRP does not support reading of (one or more) selected attributes of an MO instance in one operation.

Cons:

- Requires definition of a new class and "sub-tree structure" (of all NSV instances) for the IRPManager to process before the version information of all MOs can be identified. However, there should normally be a limited number of these instances (if there are few deviations from the default namespace-versions).

Pros/cons for Option 2b:

Pros:

- Option 2b like 2a reduces the data amount significantly. That also means reduced load on the IRPManager, IRPAgent and network in access operations over Itf-N.
- Reuses the existing structure of InventoryUnit instances, i.e. no need to build up a new containment structure to convey the version information, thereby also easier for the IRPManager to "find" it.

Cons:

- Does not work with the (currently existing) limitation that Basic CM IRP does not support reading of (one or more) selected attributes of an MO instance in one operation (the only supported semantics is "read all attributes in one shot").
- Requires support of Inventory NRM IRP for the version handling to be supported.
- Requires a new solution in the Inventory NRM for the issues mentioned above, mainly that the InventoryUnit today cannot have an association to all IOCs in the supported NRM IRPs – only to ManagedFunction IOCs, and practically only to those contained by a ManagedElement, as InventoryUnit is contained by ManagedElement.
- Requires InventoryUnit instances to be created for at least each MO where a deviation from the default namespace and version exists, even if there is no other inventory information available or relevant for that MO (e.g. a cell). This could mean the need to build a larger "inventory tree" than before, for existing IRPManagers and IRPAgents supporting the Inventory NRM.
- Not backward compatible for existing Inventory NRM implementations.

Pros/cons for Option 3:

Pros:

- Backward compatible in that no existing IOCs (including Inventory NRM) are modified.
- A solution that does not require any modification or enhancement of NRM structure.
- Works independently of the (currently existing) limitation that Basic CM IRP does not support reading of (one or more) selected attributes of an MO instance in one operation.
- Option 3, like 2a and 2b, reduces the data amount significantly. That also means reduced load on the IRPManager, IRPAgent and network in access operations over Itf-N.
- Reuses the existing structure of InventoryUnit instances, i.e. no need to build up a new containment structure to convey the version information, thereby also easier for the IRPManager to "find" it.

Cons:

- Requires definition of a new operation.

7.2.4 UC14 (PM file format)

TBD

7.2.5 UC15 (measurement types)

TBD

7.2.6 UC17 (SOA information)

TBD

7.3 Common function (operation) for all the supported version handling capabilities

7.3.1 Pros and cons

Advantages with one common solution:

- One common and consistent function, less error-prone and easy to use for IRPManagers

Disadvantages with one common solution:

- More redundancy; all parameters of the function are not relevant to all requests.
- Lots of initial work in standardisation as well as implementations to redefine many existing capabilities
- Not backward compatible.

7.3.2 Proposed solution

TBD

7.4 Protocol(s)

TBD

8 Recommendations

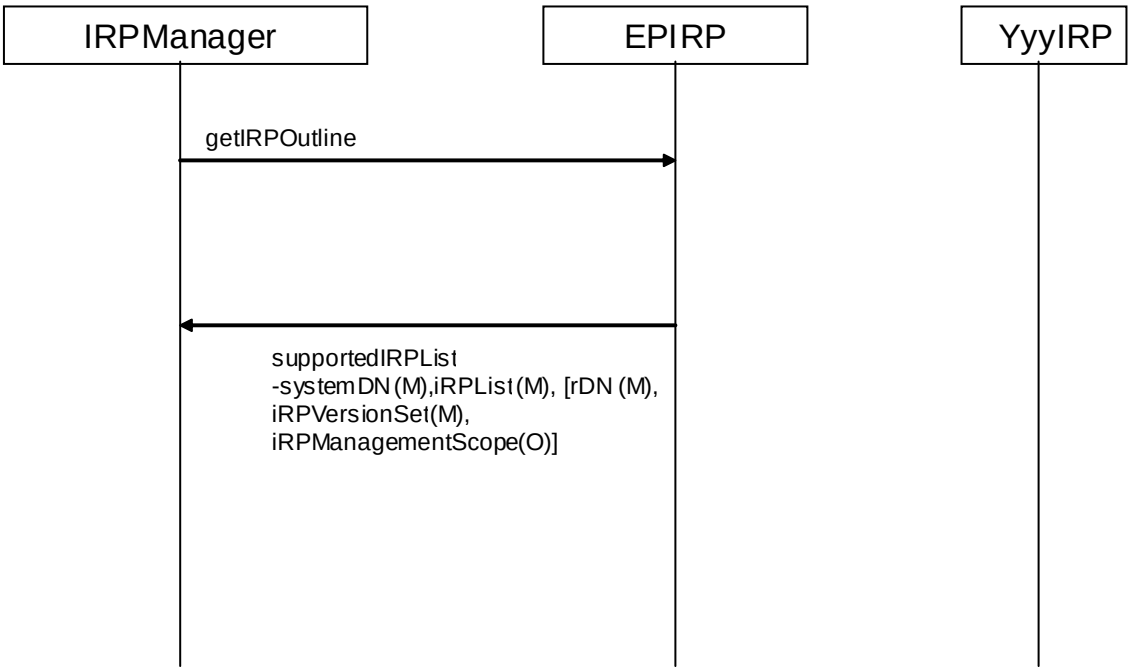
It is recommended that the alternative solutions to Use Case 13, as described in subclause 7.2.3 of this TR, are used as a basis for relevant Change Requests to specify a standardised solution for Use Case 13 over Itf-N. It is also recommended that a final solution is applicable to Converged Management.

Annex A:
Use Case Sequence Diagrams

This annex contains Sequence Diagrams for the Use Cases of clause 6.1.

USE CASE 1

IRPManager requests a list of all supported interface IRPs and versions

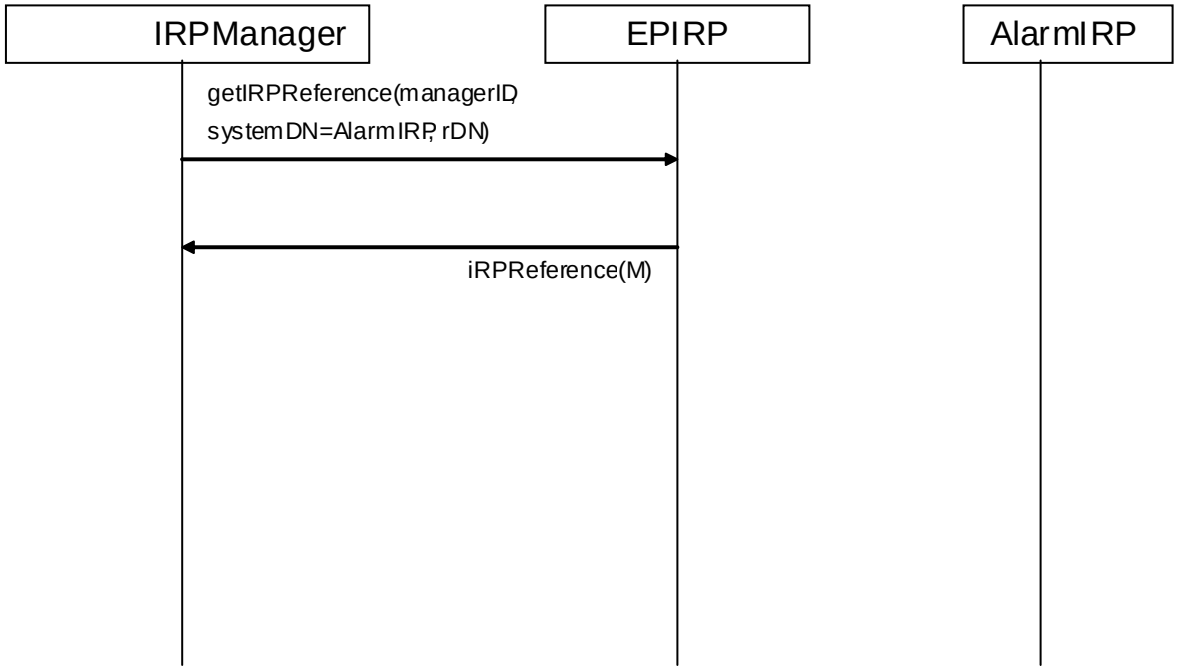


- IRPManager contacts EPIRP using the getIRPOutline operation.
- EPRIP responds with the supportedIRPList that contains the list of all IRPs and versions. The management scope of each is optionally available in the response.

The current system supports the discovery of the IRPVersions.

USE CASE 2 – Get IRP Reference.

IRPManager requests the IRP Reference

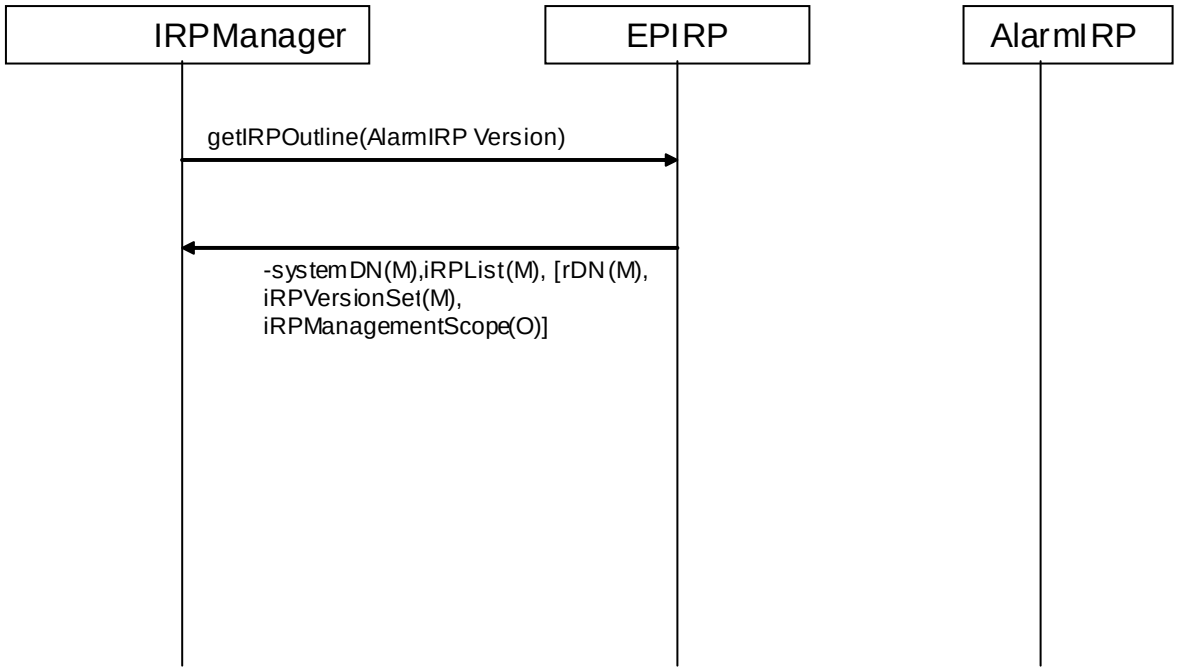


The IRPManager contacts the EPIRP using the getIRPReference request.

The EPIRP responds with the iRPReference.

USE CASE 3a - Using EPIRP

IRPManager request a list of supported Alarm IRP versions



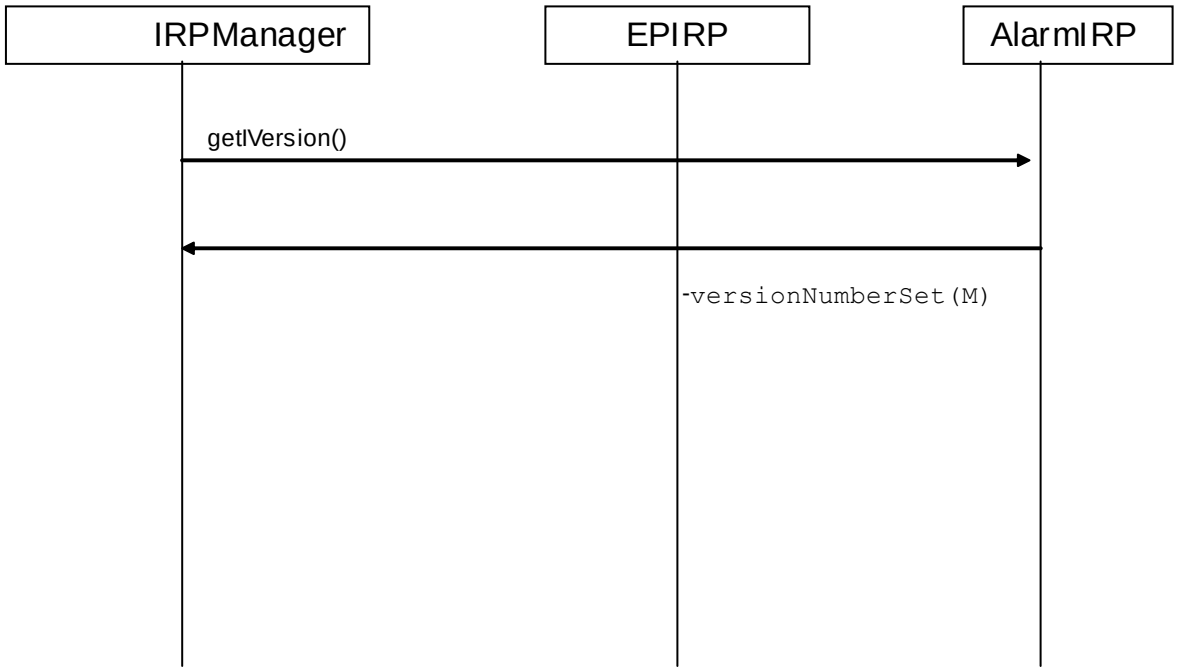
- IRPManager contacts EPIRP with the getIRPOutline with an input parameter (Alarm IRP Version)
- EPIRP responds with the systemDN and iRPList, rDN and iRPVersion

The getIRReference is used to list one instance of the Alarm IRP. The IRPManager must first obtain the complete list to find one instance.

Dependency: Requires use case #1.

USE CASE 3b – Directly to AlarmIRP

IRPManager requests a list of supported Alarm IRP versions



- IRPManager directly contacts AlarmIRP with the getIRPVersion
- AlarmIRP responds with the supported AlarmIRP SS version numbers

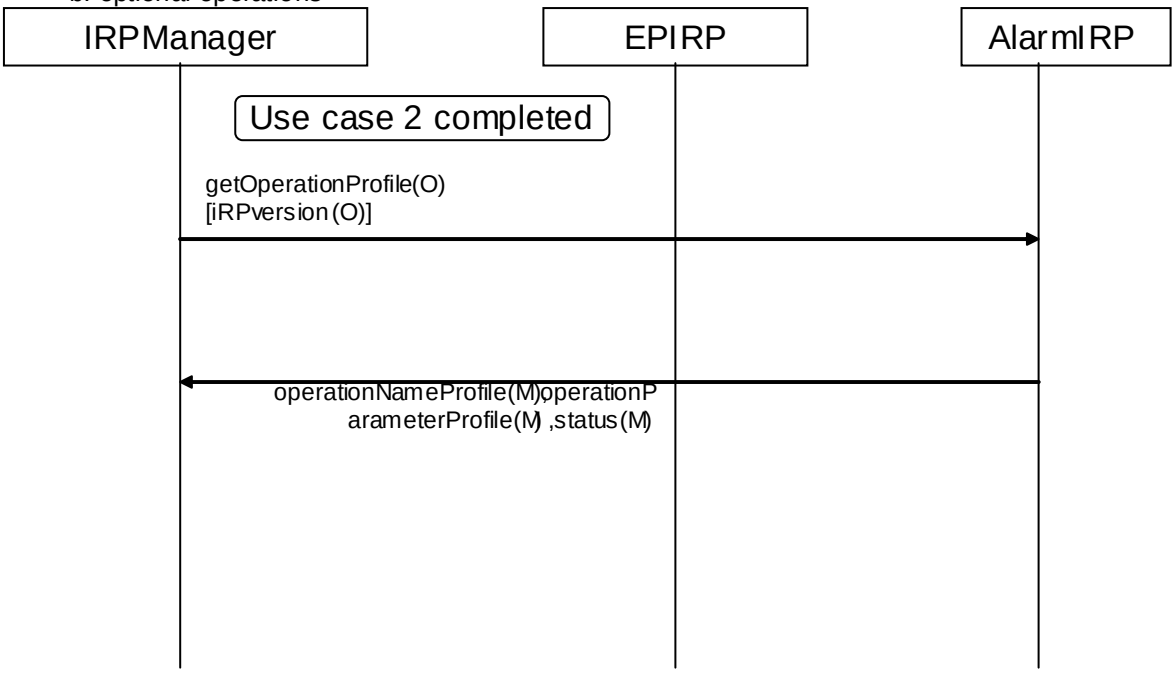
Dependency: Requires use case #1.

USE CASE 4

IRPManager requests a list of all supported Alarm IRP operations

a. mandatory operations

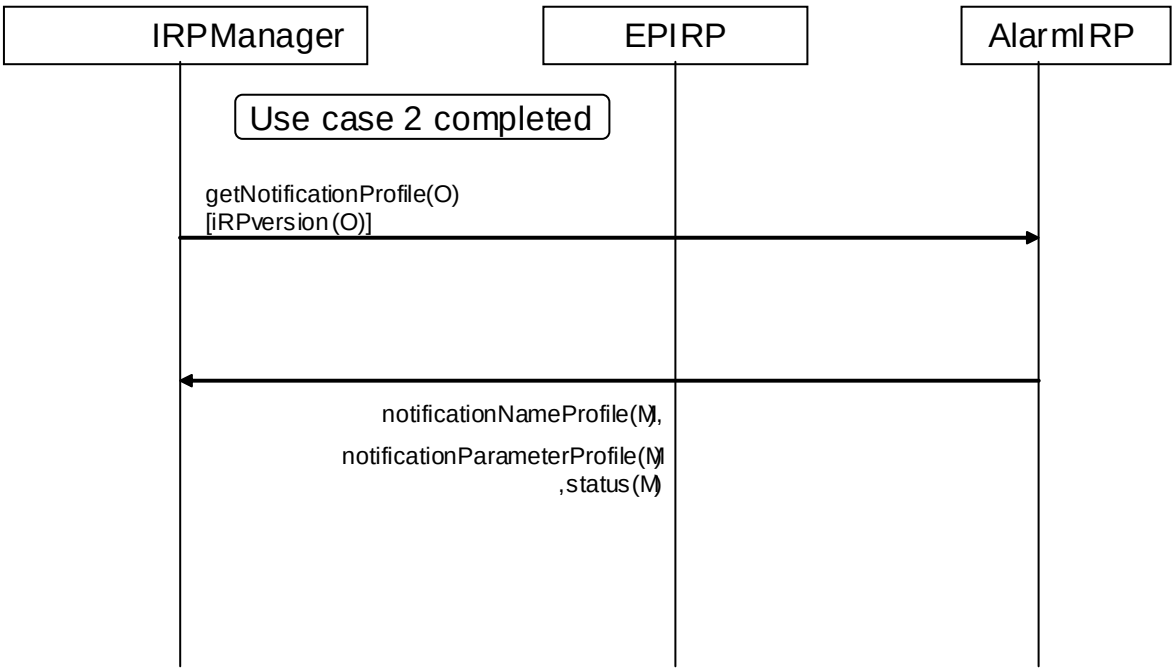
b. optional operations



- Prerequisite steps: Use case 2 completed.
- IRPManager contacts the Alarm IRP using the getOperationProfile request.
- Alarm IRP responds with the operationNameProfile containing all supported operations.
  - The Alarm IRP only responds with **supported operations,** no indication is given if the operation is mandatory or optional.

USE CASE 5

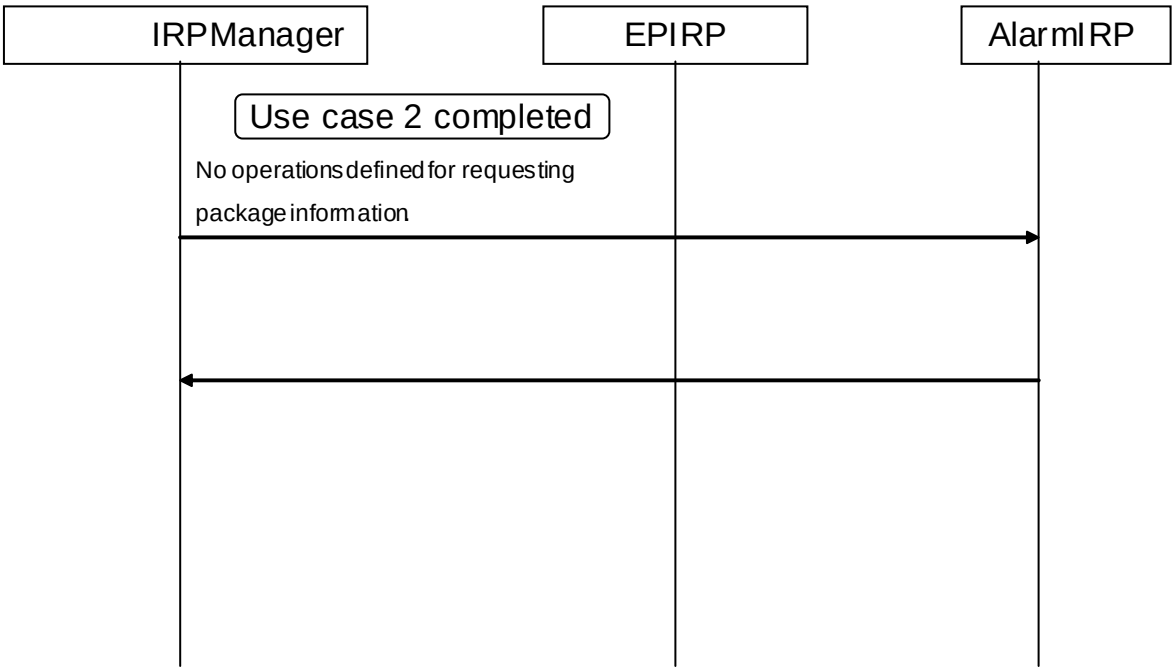
IRPManager requests a list of all supported Alarm IRP notifications (M/O)



- Prerequisite steps: Use case 2 completed.
- IRPManager contacts the Alarm IRP using the getNotificationProfile request.
- The Alarm IRP responds with the notificationNameProfile containing the list of supported notifications. No indication is given if the notification is mandatory or optional.

USE CASE 6

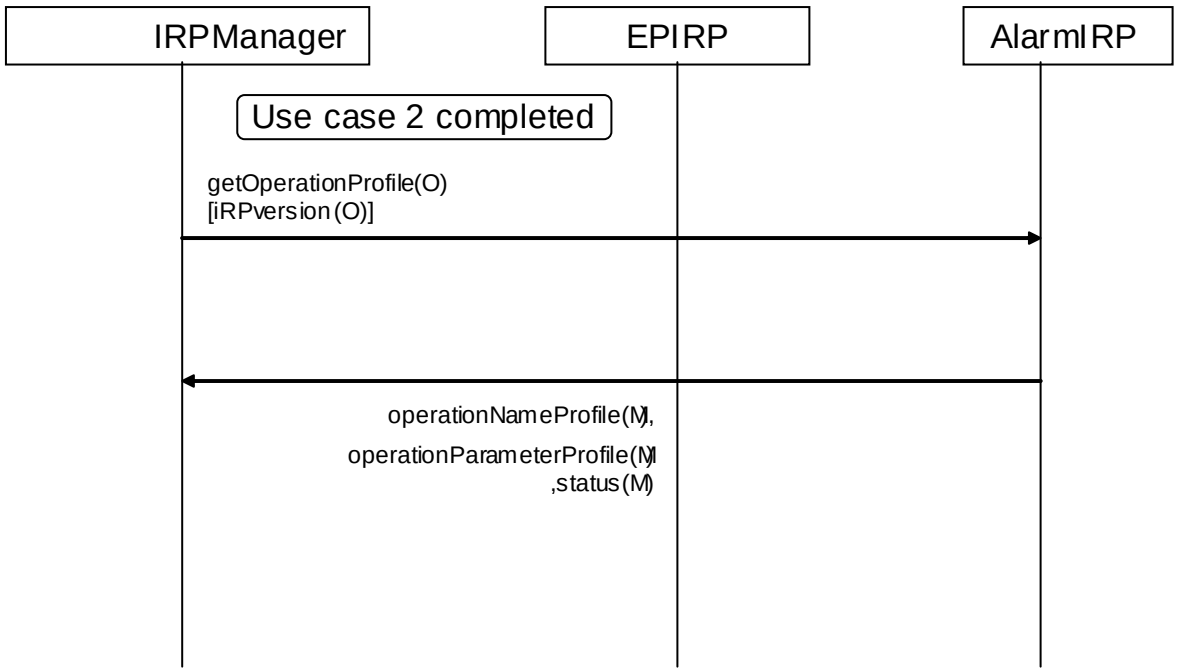
IRPManager requests a list of all supported AlarmIRP packages (M/O)



- Prerequisite steps: Use case 2 completed.
- No further steps possible, there is no method to list specific package content.
- The ability to retrieve information at the package level is not supported.,

USE CASE 7

IRPManager requests a list of all supported Alarm IRP parameters for getAlarm operation (M/O)



- Prerequisite steps: Use case 2 completed.
- IRPManager contacts Alarm IRP with getOperationProfile request
- The Alarm IRP responds with the operationNameProfile and operationParameterProfile. the Parameter profile contains the supported list of IRP paramters for all, not just the "getAlarm".
- There is no way to discover the supported output parameters.

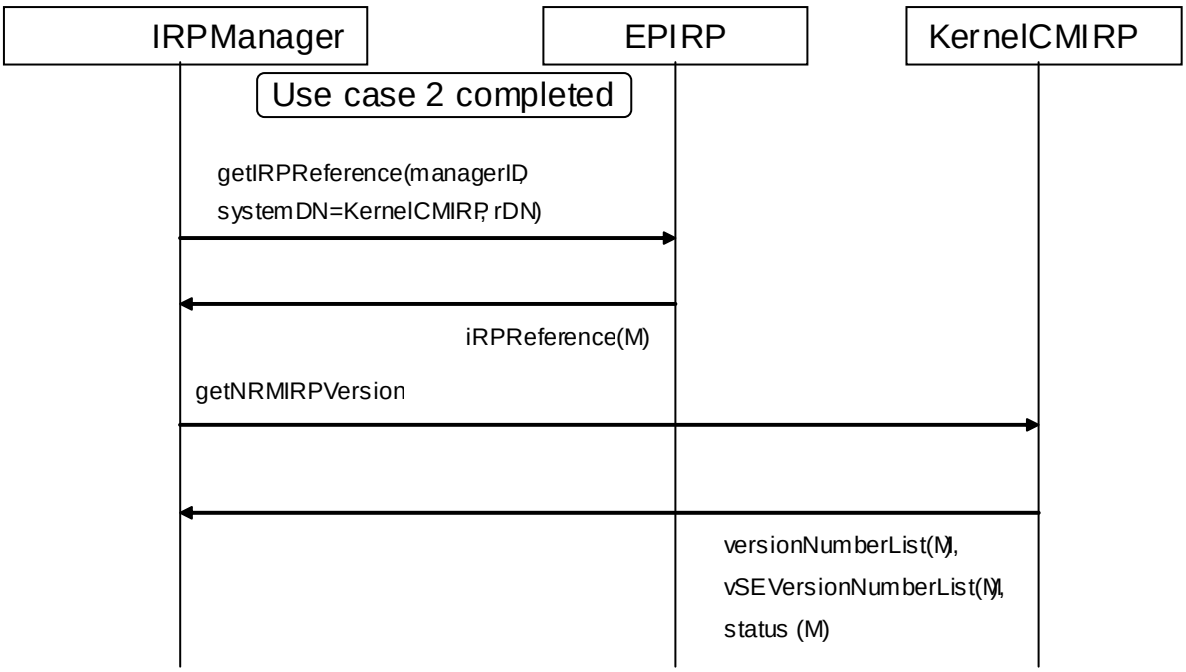
USE CASE 8

IRPManager requests a list of supported parameters for notifications of Interface IRP (e.g. AlarmIRP).

- Similar to use case 6 with notification parameters deducted from the returned list
  - With the exception: getNotificationProfile as the request and list of single NotificationProfiles in the response.

USE CASE 9

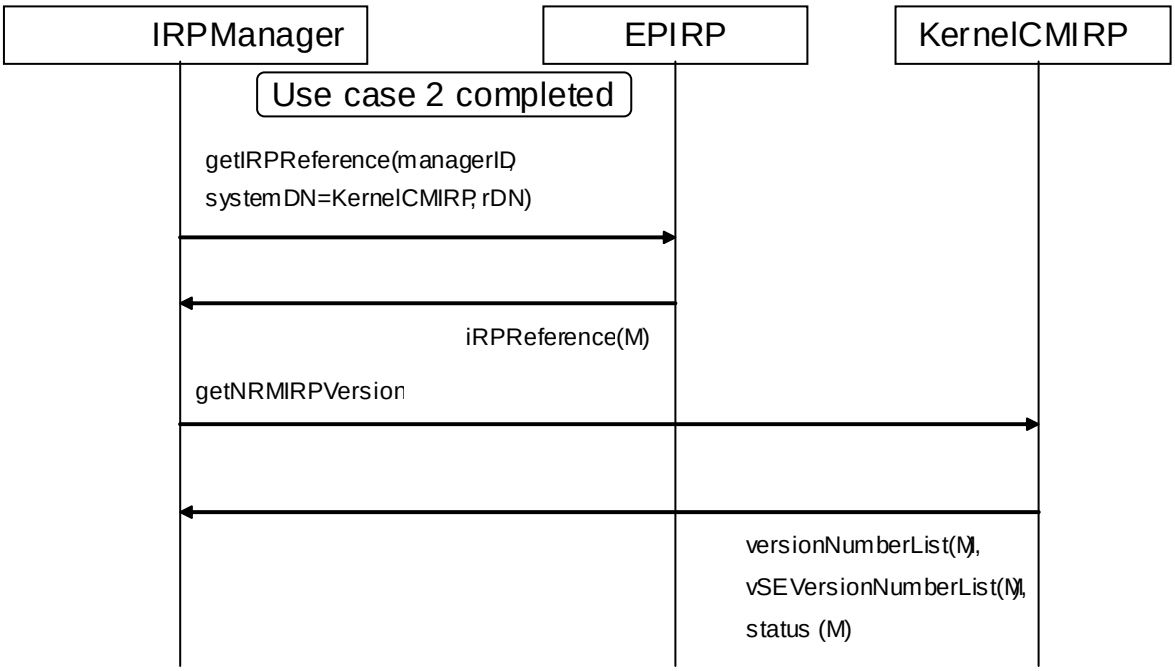
IRPManager requests a list of all supported NRMs and their versions



- Prerequisites: Use case 2-1 completed.
- IRPManager contacts EPIRP with getIRPReference request, the managerID and systemDN with a setting "KernelCMIRP" and rDn are passed.
- EPRIP returns the iRPReference
- IRPManager uses the reference (the Kernel CM IRP) to call the Kernel CM IRP with the request getNRMIRPVersion
- The Kernel CM IRP responds with the versionNumberList.

USE CASE 10

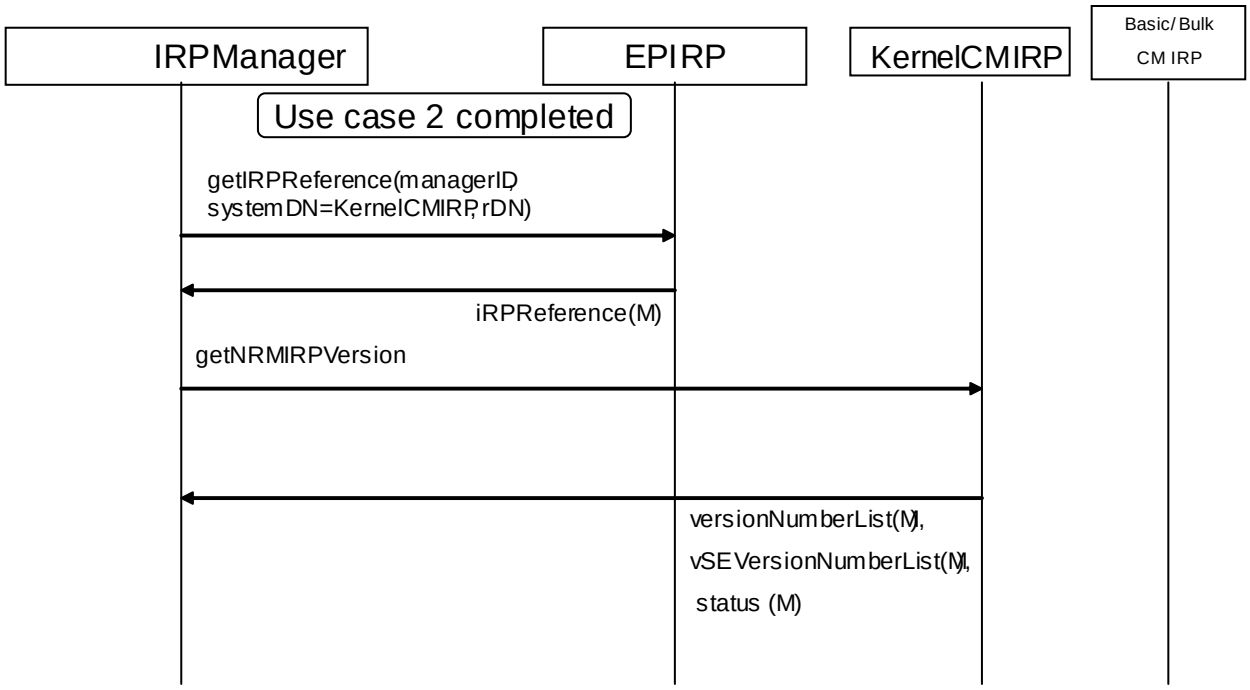
IRPManager requests a list of all supported UTRAN NRM versions



- As use case 8 sequence.
- Note: It is not possible to specify "UTRAN" - The complete list of versions is returned, the IRPManager must therefore search the list / discover the versions specific to "UTRAN"

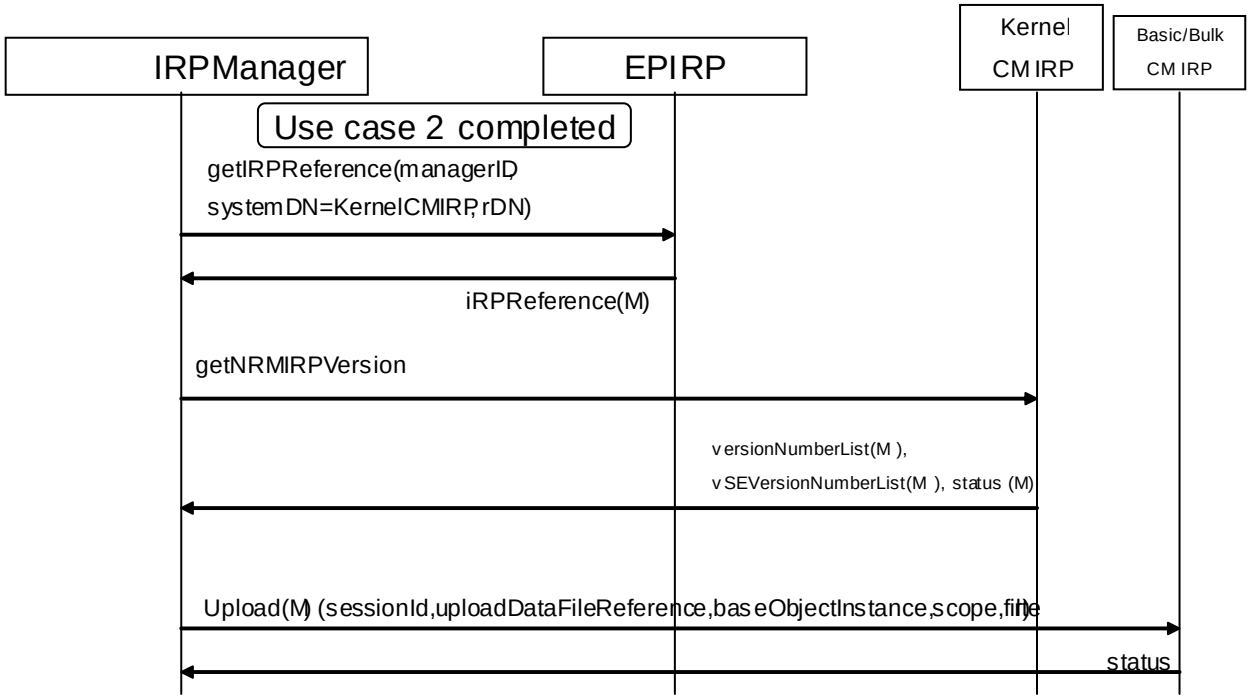
USE CASE 11-1

IRPManager requests a list of all supported UTRAN NRM objects



USE CASE 11-2

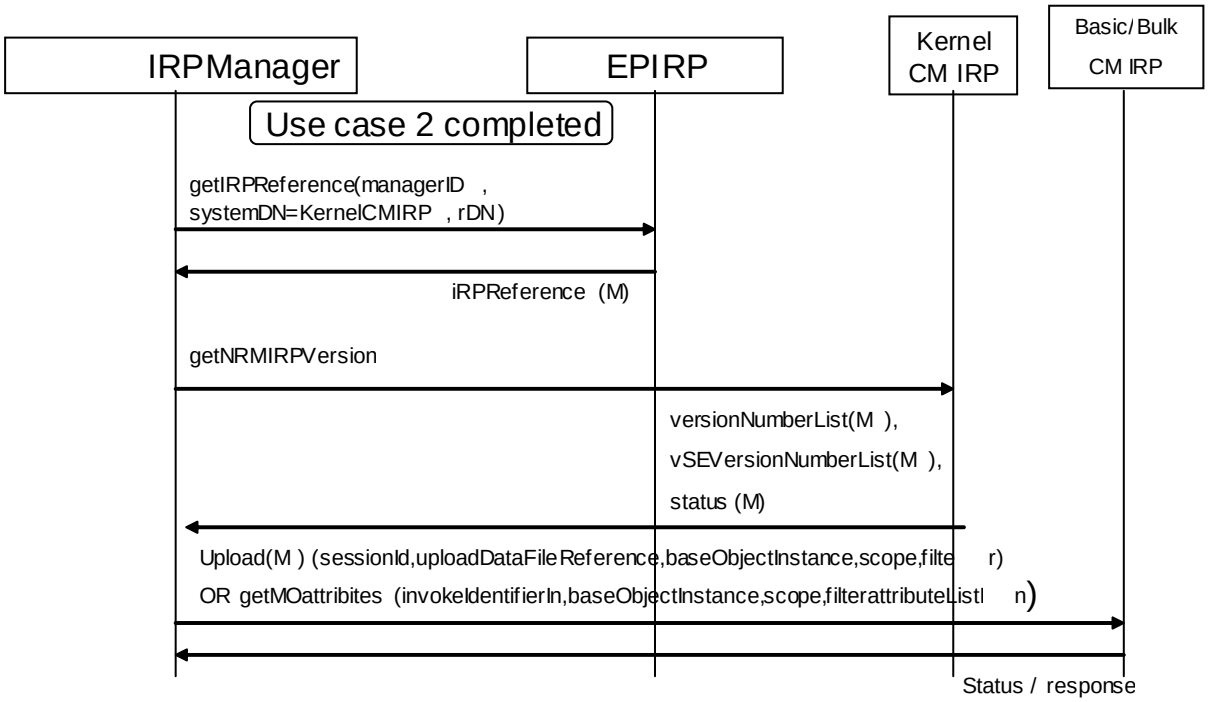
IRPManager requests a list of all supported UTRAN NRM objects



- There is no operation by which the IRPManager can obtain the UTRAN NRM Objects.
- Note: The IRPManager is able to retrieve UTRAN NRM instance information using the BasicCM getContainment operation. The response may not contain all the supported objects.

USE CASE 12

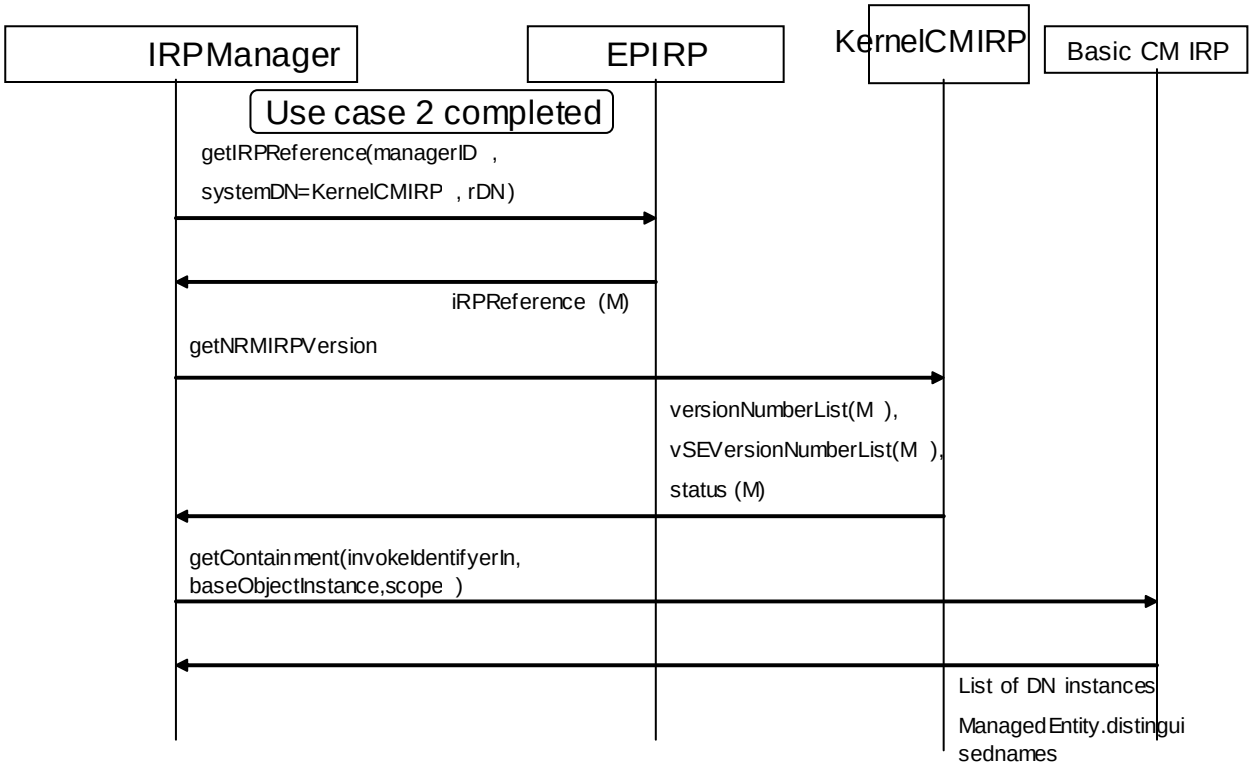
IRPManager requests a list of all supported UtranCell (UTRAN NRM) attributes



- The attributes are supported when a name-value pair is returned, otherwise not supported.
- No explicit operation defined in basic/kernel/bulk CM.

USE CASE 13

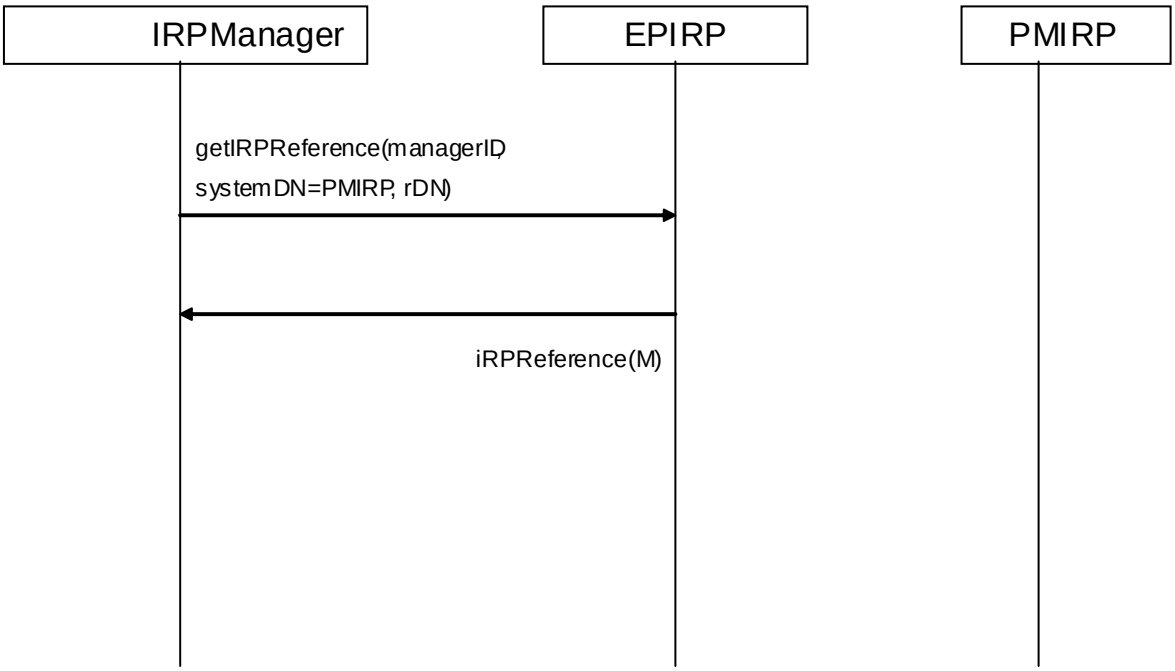
IRPManager requests a list of all supported UTRAN NRM object instances



- This use case is achieved by the getContainment operation.
- Alternatively the upload can be used.

USE CASE 14

IRPManager requests a list of all supported PM measurement types



- Not supported.

Annex B:
Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
2010-12	SP-50				Submitted to SA#50 for Information	0.3.0	1.0.0
2011-03					Updated with agreed pCR S5-111305 at SA5#76.	1.0.0	1.1.0
2011-11					Updated with agreed pCR S5-113848 at SA5#80.	1.1.0	1.2.0
2012-02					Updated with agreed pCR S5-120098 at SA5#81.	1.2.0	1.3.0
2012-08					Updated with agreed pCR S5-122150 at SA5#84.	1.3.0	1.4.0
2012-10					Updated with agreed pCR S5-122590 at SA5#85	1.4.0	1.5.0
2012-12	SP-58				Submitted to SA#58 for approval	1.5.0	2.0.0