



**Network Data Management – Usage  
(NDM-U)  
For  
IP-Based Services**

**Version 3.0**

**November 2, 2001**

**© 1999-2001 IPDR, Inc.**

## Preface

### Contacts

For general questions regarding this document and referrals to technical experts for detailed questions, please contact:

Chief Editor: Steve Cotton  
Cotton Management Consulting  
[scotton@compuserve.com](mailto:scotton@compuserve.com)

#### **Architecture Working Group** –

Lead: Charlie Nash  
Convergys Corporation  
[charles.nash@convergys.com](mailto:charles.nash@convergys.com)

Editor: open

#### **Business Requirements Working Group** –

Lead: Kelly Anderson  
Intrado  
[kanderson@intrado.com](mailto:kanderson@intrado.com)

Editor: Pat Walls  
TSI Telecommunication  
Services Inc.  
[pwalls@tsiconnections.com](mailto:pwalls@tsiconnections.com)

#### **Protocol Working Group** –

Lead: Jeff Meyer  
HP  
[jeffm@cup.hp.com](mailto:jeffm@cup.hp.com)

Editor: Ken Sarno  
IPDR.org  
[kensarno@ipdr.org](mailto:kensarno@ipdr.org)

### Acknowledgements

The following member companies contributed materially to the creation of this document:

#### **Charter Members**

ACE\*COMM  
Accenture  
Amdocs  
Apogee Networks  
Aptis  
AT&T  
Clarent  
Convergys Corporation  
CSG Systems, Inc.  
Daleen Technologies  
DST Innovis  
ECTel  
Empowertel  
HP  
Intrado  
IP23  
Lucent Technologies  
NARUS  
Portal  
Savera  
Sprint PCS  
Telesens  
TeleStrategies  
XACCT Technologies

#### **Supporting Members**

Abiliti Solutions  
American Management Systems  
Comptel Plc  
Cotton Management Consulting  
Intec Telecom Systems  
MetraTech Corp.  
Openet Telecom LTD  
Solect Technology Group  
Telcordia Technologies  
Virtual Summit, Inc.

#### **Associate Members**

BusinessEdge Solutions  
Filetek, Inc.  
InformationView  
Microsoft  
MIND CTI  
Nexus Telecom AG  
RateIntegration  
Telution, Inc.  
TSI Telecommunications  
Services, Inc.

---

## Abstract

In order to develop the IPDR record format it is required that a framework be in place that formally classifies the “IP network and service elements” and “support systems.” Next, the relationship between such subsystems needs to be in place to determine the flow of information between components. Finally, the requirements of each subsystem must be determined in order to specify the type of “IP resource and service usage information” that will be exchanged.

This document proposes a reference model to satisfy the above requirements. Chapter 2 proposes a general IP support system framework. This Chapter is intended to identify key components found in production IP networks, present the relationship between them, and define basic terminology. The Chapter focuses on the network and service element layer (NSE), and illustrates the concepts with example ISP network infrastructures. Chapter 3 details the business requirements that the structures and systems defined in Chapter 2 must satisfy. Next, Chapter 4 describes the information flow requirements between the NSE layer and the various support systems identified.

The following summarizes the essential content of each revision of this document, as of that release:

- 1.0** – This revision of the document represents the state of the work at a point determined by the working groups to be useful for broader review and validation. Many issues have been identified for further work and are not fully addressed in this issue. Additionally, it is anticipated that domain experts will contribute via liaison relationships currently being established. The services represented in this version may be substantially altered once these contributions are considered, even if at the expense of backward compatibility. Future revisions are expected to make every attempt to preserve investments made by service providers and solution vendors by considering backward and forward compatibility whenever it is practical.
- 1.1** – This revision was editorial, reflecting comments received from public review and experiences of the first prototype implementations. No significant changes to the 1.0 content occurred.
- 2.0** – This revision also introduces a major upgrade of the syntax notation of the protocol, namely XML Schema versus XML 1.0. This upgrade has been introduced to allow the protocol to specify strong typing of the usage attributes, thus conforming to the business requirements for data integrity. In addition, the dynamic operation of IDPR document transport has been specified, using the consensus choice for best conforming to business requirements, Simple Object Access Protocol (SOAP). Finally, the usage attributes for each of the services defined in the Business Requirements chapter are now formally specified, using the XML Schema definition supplied in the Protocol chapter.
- 2.5** – This revision separates the service-specific requirements and associated formal specifications into individual documents, thus allowing for addition and modifications on a per-service basis without requiring a re-issue of this document. Also, the early results of the recently constituted Architecture Working Group are reflected in Chapter 2. Finally, substantial upgrade of the transport protocol is reflected in Chapter 4.
- 2.6** -- This revision expands the discussion of the architecture to differentiate the interfaces and nodes of the reference model. The operations model has been moved to Chapter 2 and represented as individual scenario variations. The file transfer protocol has been enhanced to bring it in alignment with the generic requirements for revenue assurance. Finally, the capabilities file has been defined and a schema describing it is included.
- 3.0** -- This revision adds the compact encoding of the IPDRDoc. This encoding, based on XDR, is defined to address the operational efficiency of the NDM-U protocol in the dimensions of storage, transmission time, and processing overhead. A preliminary SOAP mapping was specified in NDM-U versions 2.0, 2.5 and 2.6. It has been removed from version 3.0. This decision was based on the lack of implementation and operational experience. A full mapping of the operations defined in this section are an area for further study.

---

## Change History

1.0 WDA	12/14/1999 Working Draft Revision A - Circulated to Membership for Comment
1.0.WDB	1/17/2000 Working Draft Revision B - Circulated to Membership for Comment
1.0 RDA	3/6/2000 Review Draft Revision A – Circulated to Steering Committee for Comment
1.0	3/20/2000 Steering Committee Approval
1.0.1	6/5/2000 Interim Issue to Correct XML Example
1.1 RDA	6/12/2000 Review Draft of 1.1
1.1	6/26/2000
2.0 RDA	September 15, 2000 Review Draft A of 2.0
2.0 RDB	September 22, 2000 Review Draft B of 2.0
2.0 AD	October 1, 2000 Approval Draft to be Balloted at St. Louis Meeting
2.0	October 23, 2000 Official Release
2.5 RD A	February 19, 2001 Review Draft A
2.5 RD B	March 6, 2001 Review Draft B
2.5 Approval	March 12, 2001 – Steering Committee Conditional Approval
2.5	April 12, 2001 – Production Release
2.6 RD A	June 11, 2001– Review Draft A
2.6 RD B	June 18, 2001 – Review Draft B
2.6 BD	June 25 – Ballot Draft
2.6	June 28, 2001 – Steering Committee Approval
3.0 ID	<b>November 2, 2001</b> – Initial Draft for Membership Review
3.0 BD	October 22, 2001 – Ballot Draft for Steering Committee Approval
3.0	November 2, 2001 – Production Release

## Table of Contents

Preface.....	2
Contacts .....	2
Acknowledgements .....	2
Abstract.....	3
Change History .....	4
1. Introduction .....	7
1.1 Purpose.....	7
1.2 Scope.....	7
1.3 Compatibility.....	7
1.4 Timeline .....	7
1.5 References.....	7
1.6 Overview.....	8
1.7 Terminology and Glossary .....	9
2. IPDR Reference Model.....	11
2.1 IPDR and the TMF Model.....	11
2.2 IPDR NDM-U High-level Model.....	11
2.3 IPDR Record Contents .....	13
2.4 The NDM-U Reference Model.....	14
2.4.1 NDM-U Nodes.....	15
2.4.2 NDM-U Interfaces .....	16
2.5 NDM-U Model Usage Scenarios.....	17
2.5.1 SE Directly Interfaced to BSS .....	18
2.5.2 SE Interfaced to BSS Via Mediation System.....	18
2.5.3 SE Interfaced to BSS Via Multiple Mediation Devices .....	19
2.5.4 Simple Roaming with Separate Access and Home Service Providers.....	19
2.5.5 Service Provider also Provides an Application Service.....	20
2.5.6 Separate Home, Access, Transport, and Application Service Providers Acting in Concert .....	21
3. Business Requirements.....	24
3.1 Introduction .....	24
3.2 Assumptions.....	24
3.3 Generic Requirements.....	24
3.3.1 Mediation.....	24
3.3.2 Format.....	25
3.3.3 Application Protocol.....	25
3.3.4 Usage Attributes .....	25
3.3.5 Settlement .....	25
3.4 Listing of Services.....	25
3.4.1 Services Covered .....	25
3.4.2 Services for Future Consideration .....	26
3.4.3 Services Considered by other Organizations .....	26
4. NDM-U Protocol – IPDR Document Structure and Transfer.....	27
4.1 Overview of Encoding.....	27
4.2 XML Encoding.....	27
4.2.1 IPDR Master Schema.....	28
4.2.2 Annotated IPDR Master Schema .....	32
4.3 Compact Encoding .....	35
4.4 Document Transfer.....	51
4.4.1 Introduction.....	51
4.4.2 Groups, Sequence Numbers, Document Ids and Subscriptions .....	52
4.4.3 Protocol Primitives and Parameters.....	54
4.4.4 State Diagrams .....	60
4.4.5 State Definitions .....	64

---

4.4.6	Stimulus and Responses.....	65
4.4.7	Capability Files and Protocol Extension.....	65
4.4.8	File Sharing Protocol Mappings .....	68
4.4.9	Security Considerations .....	74
4.4.10	Implementation Considerations .....	74

# 1. Introduction

## 1.1 Purpose

This document, in conjunction with the referenced Service Definition documents, is intended to specify technical information that is sufficient for practical implementations of interchange of usage data among service elements participating in the delivery of IP-based services, either within a single enterprise or across multiple enterprises.

The IPDR organization intends to submit this specification to selected accredited organizations for consideration as an approved standard.

## 1.2 Scope

This document is limited to the discussion of issues as defined by the mission statement of IPDR.org, namely:

The IPDR Organization (the "Organization") is organized and operates as a non-stock not for profit organization for the following purposes:

- To create and promote the adoption of interoperability standards for exchanging service usage and control information between IP network or hosting elements and operations or business support systems.
- To provide a standardized framework for the development of carrier-grade support systems that enable next-generation digital service providers to operate efficiently and cost effectively.

## 1.3 Compatibility

Future revisions are expected to make every attempt to preserve investments made by service providers and solution vendors by considering backward and forward compatibility whenever it is practical.

## 1.4 Timeline

## 1.5 References

- [1] Telecom Operations Map - Version 2.1, GB910, *TeleManagement Forum* (TMF - <http://www.tmforum.org>), March 2000.
- [2] XML Schema Part 1: Structures, W3C Candidate Recommendation 24 October 2000. See <http://www.w3.org/TR/xmlschema-0>.
- [3] XML Schema Part 2: Data Types, W3C Candidate Recommendation 24 October 2000.
- [4] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. See <http://www.w3.org/TR/SOAP>
- [5] OSF CAE Specification, Document C706, 1997, Appendix A, located at: <http://www.opengroup.org/onlinepubs/009629399/>.

- [6] IPDR Organization Home Page: <http://www.ipdr.org>.
- [7] Specification Guide for NDM-U Services – 1.0, October ??, 2001, IPDR.org.
- [8] Application Service Provider Service Specification – ASP 3.0-A.0
- [9] Voice over IP Service Specification – VoIP 3.0-A.0
- [10] Electronic Mail Service Specification – E-mail 3.0-A.0
- [11] Authentication and Authorization Service Specification – A&A 3.0-A.0
- [12] Internet Access Service Specification – GPRS and WAP 3.0-A.0
- [13] Wholesale Service Specification 3.0-A.0
- [14] Streaming Media Service Specification – VoD 3.0-A.0

## **1.6 Overview**

This specification is divided into three major chapters:

- IPDR Reference Model - a definition of the abstract and operational relationships between entities involved in the generation, recording, storage, transport, and processing of usage attributes.
- Business Requirements - a definition of business requirements to be addressed by the protocol specification and specific scenarios for the major process flows anticipated in actual application.
- Protocol - the notation, data unit syntax, and dynamic procedures involved in the operation of the interfaces specified in the reference model.

The Protocol chapter represents the specific design produced through analysis of the Business Requirements chapter, consistent with the Reference Model chapter.

## 1.7 Terminology and Glossary

### 1.7.1 Terminology

Term	Definition
Accounting	The process of collecting and analyzing <b>service</b> and <b>resource usage</b> metrics for the purposes of capacity and trend analysis, cost allocation, auditing, and billing, etc. Accounting management requires that resource consumption be measured, rated, assigned, and communicated between appropriate business entities.
Mediation	In view of network reference model, Mediation refers to the combination of the logical entities IPDR recorder, IPDR transmitter, and IPDR store.
Resource	A quantifiable asset employed by a <b>Service Provider</b> , or on behalf of a <b>Service Provider</b> by another Service Provider, to fulfill a request of a <b>Service Consumer</b> . (Examples include: files, communications, goods, etc).
Roaming	Service usage initiated by a service consumer and provided by a service provider other than the one with which the service consumer have business relationship.
Service	Network and/or application operation that provides the <b>Service Consumer</b> with the requested <b>resource</b> .
Service Consumer	The beneficiary (human or system) of a <b>service</b> .
Service Element	Any element that is responsible for fulfilling a <b>Service Consumer</b> request. (Examples include: network equipment and system processes)
Service Provider	An enterprise that provides communications-based <b>services</b> .
Session	A set of related service usages; service usages may or may not be time based in the unit of measurement.
Usage	Consumption of <b>resources</b> and <b>services</b> by a <b>Service Consumer</b> .
Usage Attribute	A parameter whose value indicates some aspect of <b>usage</b> of a given <b>service</b> and/or <b>resource</b> .
Usage Entry <sup>1</sup>	A <b>Service</b> -specific trigger resulting in the generation by a <b>Service Element</b> of a set of <b>Usage Attribute</b> values related to <b>Usage</b> specific to a given <b>Service Consumer</b>

<sup>1</sup> Because of legacy issues, a Usage Entry from a given Service Element will not initially conform to an IPDR specification or, in some cases, may never conform. To be considered a Usage Entry the information presented or made available by inference from the Service Element must minimally contain attributes from some of the general attribute categories.

## 1.7.2 Glossary:

ANI	- Automatic Number Identification
ASP	- Application Service Provider
BSS	- Business Support Systems
CCI	- Call Clarity Index
CRM	- Customer Relationship Management
DSS	- Decision Support Systems
DTD	- Document Type Definition
DSL	- Digital Subscriber Line
EP	- End Point
ESN	- Electronic Serial Number
ETSI	- European Telecommunications Standardization Institute
FoIP	- Fax over IP
GK	- Gate Keeper
GPRS	- General Packet Radio Service
GSM	- Global System for Mobile Communications
IETF	- Internet Engineering Task Force
IMSI	- International Mobile Subscriber Identity
IP	- Internet Protocol
IS	- IPDR Store
ISDN	- Integrated Services Digital Network
ISO	- International Standardization Organization
ISP	- Internet Service Provider
IT	- IPDR Transmitter
ITU-T	- International Telecommunications Union – Telecommunications Standardization Section
MOS	- Mean Opinion Score
NDM	- Network Data Management
NDM-U	- Network Data Management - Usage
NSE	- Network Service Element
OSS	- Operations Support System
PLMN	- Public Land Mobile Network
PSTN	- Public Switched Telephone Network
QoS	- Quality of Service
RADIUS	- Remote Access Dial-In Usage Server
RAS	- Remote Access Server
SC	- Service Consumer
SCN	- Switched Communications Network
SE	- Service Element
SMS	- Short Message Service
SP	- Service Provider
TIPHON	- Telecommunications and IP Harmonization over Networks
TMF	- TeleManagement Forum
TOM	- Telecommunications Operations Map
UA	- Usage Aggregators
UC	- Usage Collectors
VoIP	- Voice over IP
VPN	- Virtual Private Network
WAP	- Wireless Application Protocol
xDSL	- Digital Subscriber Line of type x
XML	- eXtensible Markup Language

## 2. IPDR Reference Model

The IPDR organization has adopted the Telecommunication Management Forum's (TMF) telecommunications operation map (TOM) for the purposes of motivating the functional role and interfaces of the IPDR specifications relative to operations support systems (OSS). We have chosen the TOM because it is a well-known, industry-accepted organizational model of telecommunications operations business processes used by carriers and service providers today. The TMF Model is useful as a model of typical systems, and as motivation for design decisions. However, the TMF Model itself is not part of IPDR, and the data structures and interfaces of IPDR may be used in systems that vary substantially from the TMF Model. See [1] for more details.

### 2.1 IPDR and the TMF Model

The TOM, shown in Figure 1, identifies the core operation support processes found in a production carrier business operation. The systems that implement the customer care, services development/operations and network/systems management processes each provide a well-defined set of services that enable a carrier to successfully deploy and manage telecommunications services. As the model shows, these systems are organized in a layered fashion. Thus, each component builds on the services provided at a lower layer (and possibly adjacent components) to deliver the required functionality. The IPDR organization's charter is to facilitate the integration of IP-based network elements into billing, reporting and assurance systems. In particular, one key goal is to define a common usage record format and exchange protocol to facilitate the flow of usage information from IP network elements managers to support systems. In the TOM, the network data management (NDM) component (defined as part of the network and systems management processes) defines the device-independent collection mechanism for such purposes. As such, the work of this specification falls primarily within the domain of the Network Data Management component.

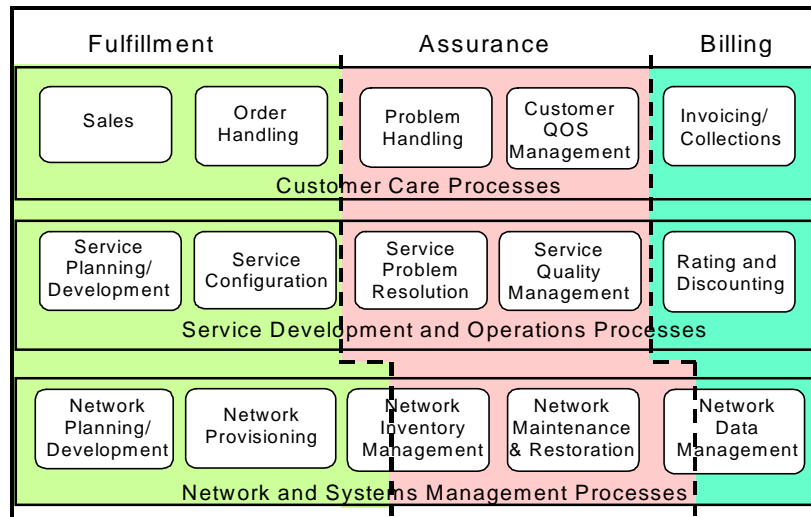


Figure 1

Telecommunications Operations Map (TOM)

### 2.2 IPDR NDM-U High-level Model

Within the scope of the NDM module, the TOM assumes interfaces to billing (i.e., rating and discounting) and customer care systems (i.e., service quality management and problem resolution). Likewise, the TOM

assumes that the NDM component must interface directly with the network and service element manager to accomplish their various services.

The IPDR NDM-U reference model, shown in Figure 2, expands on the NDM definition by dividing the module into layers; namely: (1) the network and service element layer, (2) the mediation layer, and (3) the business support systems layer. Each layer is discussed below:

*Network and service element layer (NSE):* The NSE layer consists of all the network and service elements required to provide an IP-based service to a given customer. For example, routers, access devices and transmission facilities together provide basic connectivity; firewalls might provide a security service; email, file and print servers provide application services; gateways provide a translation service from circuit to packet voice; and more. In addition to physical devices, the systems that configure and manage such devices are considered part of the NSE layer (note, that this functionality is identified as adjacent component within the “Network and Systems Management” layer in the TOM model). Examples here include a bandwidth management system, H.323 gatekeeper, RADIUS, authentication server or network management platform.

*Mediation layer:* As shown in Figure 2, mediation systems sit between the network elements/infrastructure and the business support systems. Typically, a mediation system provides a single interface to BSS systems that provides all network usage data and often a single interface for service elements provisioning. In terms of usage collection, the goal of the mediation system is to capture all usage information required by the BSS systems, and export it within the temporal requirements. Thus, the mediation system must determine the devices at the service element layer and interface with that infrastructure to extract the relevant usage information. The second mediation goal is to pass provisioning information from the BSS, to the network elements – again, within the temporal constraints.

- ◆ *Business support systems (BSS) layer:* The BSS layer consists of the systems deployed by a Service Provider or provider to support IP business operations. This layer corresponds to the “Systems Development and Operation Processes” and “Customer Care Processes” in the TOM model. Some examples include billing (i.e., rating and discounting), customer care/relationship management, decision support, and market analysis and fraud detection. The BSS layer is the highest layer in the model. Thus, the BSS usage collection and provisioning requirements drive the mediation system and ultimately the services provided at the service element layer.

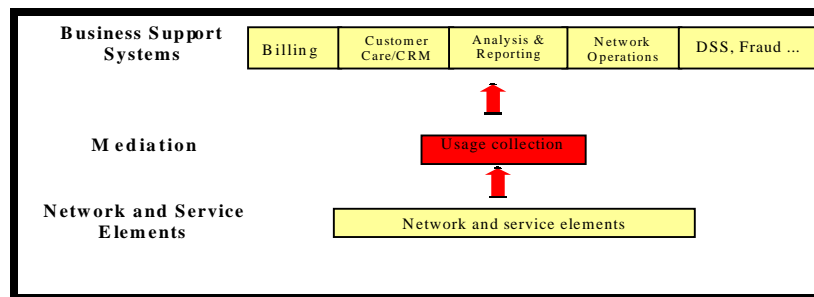


Figure 2

**IPDR NDM-U High-Level Model**

The IPDR NDM-U model shown in Figure 2 gives a layered perspective of the components and interfaces designed to meet the NDM-U specification. The usage collection process represents a flow of usage data from the network and service elements to the BSS processes. The mission statement given in the introduction limits the organization’s current scope to the usage collection path (shown flowing upward in Figure 2). Thus, provisioning or the internal design of any of the identified components is not considered in this document.

Figure 3 illustrates the usage data path from network elements (e.g., gateways, remote access servers (RAS), routers, and bandwidth managers) to a mediation device in a typical IP network scenario. Note that this example assumes the interface between the mediation device and network elements is based on a proprietary access protocol, record format and API. The mediation system aggregates, normalizes and correlates the usage data as required, and exports the data to the billing, decision support, or other business support systems.

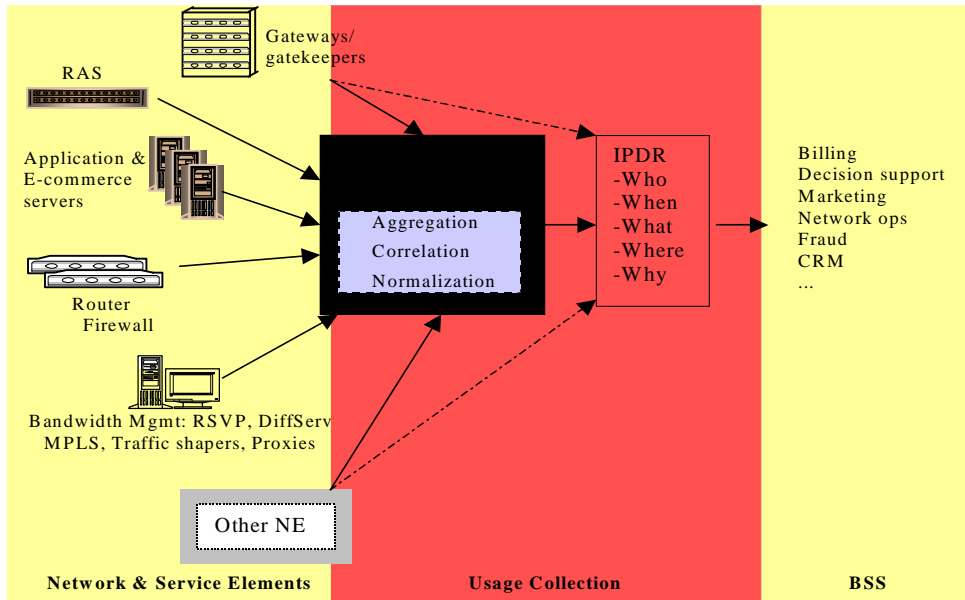


Figure 3

**IPDR Record Flow Example**

The IPDR NDM-U plays several roles in this data transfer. First, the IPDR record provides flexible structure that is sufficiently powerful to describe the usage attributes collected by the mediation system, and required by the BSS system. Second, the IPDR NDM-U provides a set of interfaces that facilitate the exchange of IPDR records between mediation systems and BSS systems, or potentially between IP network elements and BSS systems (as demonstrated by dotted lines). Finally, the IPDR specification provides a common format that facilitates the intermediate storage of IPDR records between IPDR-enabled components.

**2.3 IPDR Record Contents**

The IPDR record must be capable of characterizing any type of usage that might be collected from an IP-based network or application service. As Figure 3 identifies, there are 5 attributes common to typical IPDR records. Broadly, these components are the “who, what, where, when and why” values that describe a particular usage event. Each is described briefly below (formal definitions are provided in a later chapter):

- {Who} (Responsible for the usage)  
User ID
- {When}  
End Time or Event Time
- {What}  
Service

Usage measures / quantities  
 Ex: Bytes, packets, flows, hits, transactions, time duration...  
 QoS measures  
 State information  
 Event code (logon, logoff, threshold exceeded)  
 Other information about state transition or current state (Start Time)<sup>2</sup>

- {Where}
  - Traceability / Context
  - Source Identifier
  - Destination Identifier
  - Service Element identifier (originator)
- {Why}

Event trigger type – (i.e., why is the network and service element reporting this data?)

In addition to the “SWs” defined above, each record may include reference pointers to other IPDR records that either capture related usage information, or contain usage information that was used to create the given record.

### 2.4 The NDM-U Reference Model

In addition to the IPDR structure, the NDM-U specification defines a set of interfaces for exchanging IPDRs between NDM-U-enabled devices or systems. As will be specified in the Protocol chapter, IPDRs are packaged in protocol data units (PDUs) known as IPDR Documents (IPDRDocs). These PDUs are the entities for which protocol transactions and tracking are done. All future references to IPDR documents imply these formal protocol entities. Figure 4 shows the key interfaces and elements found within the NDM-U reference model, represented in an abstract form. For instance, a product solution might be developed that acts as the "generator"/"sensor" of usage data by means of passive monitoring of IP packet streams, packaging sets of this data into IPDRDocs for forwarding to BSSs. Such a packaging of functionality would internalize the A, B, and C interfaces, presenting to the “out side” only the D interface. Another "sensor"-only product might accept proprietary-format records of usage data from a "generator" SE that offers such records via an FTP-based A interface and offers them to BSSs via an NDM-U-compliant D interface. Both implementations are equally valid and might offer different value propositions to various SPs looking for a usage solution. Note that this model does not constrain implementations to be physically packaged as portrayed, nor to present all of the interfaces to other systems.

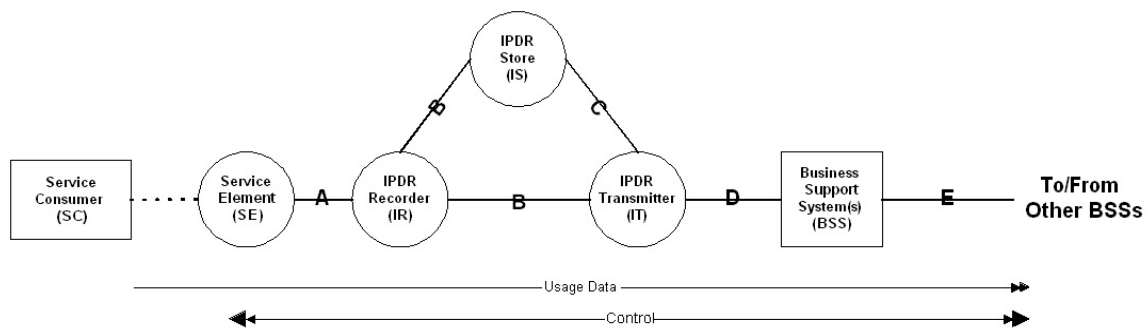


Figure 4

#### NDM-U Reference Model

<sup>2</sup> Note that "always on" services may be measured via periodic emission of IPDRs, recording usage since the last interval boundary.

This document focuses on the definition of IPDR Documents, the information content of IPDRs and the interface between ITs and BSSs. The D interface is completely specified in this document while other interfaces are identified here to aid in decomposing the problem statement.

Implementations adhering to this version of NDM-U are not required to explicitly separate the IPDR Recorder, IPDR Store and IPDR Transmitter roles, since the B and C interfaces are, as of this release, unspecified. Similarly, the E interface is not explicitly addressed in this release of the NDM-U. As of this writing, the IPDR organization working groups are beginning to address E interface. The E interface discussion will specify detailed requirements and design (at least for IPDR-IPDR exchange scenarios).

## **2.4.1 NDM-U Nodes**

### **2.4.1.1 Service Consumer (SC)**

This is the human or machine initiating requests for services from the Service Element (SE). The Service Consumer requesting and receiving services is typically the end user on end system. Note that an SC may request services from an SE via another SE that is providing access to the Internet, thus involving more than one SE in the providing of a given service.

### **2.4.1.2 Service Element (SE)**

This is the set of equipment and software that provides a valuable service to a Service Consumer. The Service Element provides access to services and resources requested, authenticates Service Consumers, authorizes access, performs accounting measurement for resources provided, provides services requested by Service Consumer, and performs accounting measurement for services provided. Many classes of Service Elements exist: Network routers, VOIP switches servers, ASP applications servers, etc. The IPDR reference model applies to any type of Service Element that is capable of generating accountable usage records (i.e. a record of which services were provided to which consumers). An SE is a generalized superset of Network Elements (NE).

### **2.4.1.3 IPDR Recorder (IR)**

This entity performs two principal functions: 1) Mediating proprietary protocols and data transactions from a Service Element; and 2) Producing IPDRs resulting from the transformation of that proprietary data. IPDR Recorder packages usage information into IPDRs, it presents a stream of IPDRs to an IPDR Store and/or an IPDR Transmitter. Multiple Service Elements can be connected to a single IPDR Recorder.

### **2.4.1.4 IPDR Store (IS)**

This is the entity that provides persistence to the IPDRs recorded by an IPDR Recorder. The IPDR Store receives IPDRs from an IPDR Recorder and packages them into IPDR documents stored in a non-volatile medium. The IS, also, provides a repository of IPDRDocs for transmission or retransmission of selected IPDRDocs by the IPDR Transmitter to one or more Business Support Systems. An IPDR Store can receive IPDRs from one or more IPDR Recorders and the IS can deliver IPDRDocs to one or more IPDR Transmitters.

### **2.4.1.5 IPDR Transmitter (IT)**

The IPDR Transmitter delivers IPDR documents to Business Support Systems. These documents may be retrieved from an IPDR Store, or they may be created directly by the IT. This entity performs three principal functions: 1) Packaging of IPDRs from the IPDR Recorder into IPDRDocs; 2) Organization of IPDRDocs containing usage of the same service type into Groups; and 3) Transmission (or retransmission) of IPDRDocs from Groups to one or more Business Support Systems, using one of a set of transfer protocols. An IPDR Transmitter can deliver IPDRDocs to one or more Business Support Systems.

### 2.4.1.6 Business Support System (BSS)

This entity is any system that implements a technical or commercial function to perform one or more processes in a telecommunications enterprise. A Business Support System receives information contained in IPDRDocs from an IPDR Transmitter, processes the information contained in the contained IPDRs for use in the commercial activities of a Service Provider, and presents information for transmittal to other Business Support Systems. A Business Support System can receive IPDRDocs from one or more IPDR Transmitters.

## 2.4.2 NDM-U Interfaces

### 2.4.2.1 A Interface

The A interface delivers usage information from Service Elements to IPDR Recorders. The IPDR does not attempt to constrain the file naming conventions, format, transfer protocol, sequencing, or other details of the A interface data transfer mechanism. It is not assumed to be real-time nor batch.

An A interface receives data generated by a Service Element. At the present state of the industry, this is typically a vendor-specific access protocol – often file-based, but sometimes CORBA, HTTP, or socket-based. It typically includes high-volume data at a highly detailed granularity. However, the data is often very equipment-level in detail (e.g. port number, IP address, or line number). Much of the A interface data may require translation before it can be related to business-level entities (e.g. customer and price plan).

In addition to being equipment-level in detail, data transferred on the A interface is often incomplete for business purposes. For example, interim usage details are common. Interim usage details are describing a service delivery event that cannot be correctly interpreted without context provided by other data. For example, “Start Call” and “Stop Call” entries must be correlated to each other in order to determine call length. Other examples of interim entries might be: Start Application, Service Query, Connect additional line to conference, Drop line from conference, Convert text message to voice, etc.

Communication at this granular level and correlation/translation into useful business-level identifiers is a core competency of mediation packages. NDM-U assumes the communication, translation, and correlation to be revenue-grade.

The IPDR Organization may comment on the data elements required by the A interface for specific services. Such comments are contained in the “Assumptions for other interfaces” section of the Service Specifications.

Not all requirements inherent to the A interface are well supported by the IPDR NDM-U standard. However, some types of equipments (e.g. application servers) are good candidates for generating usage data using IPDR format and NDM-U protocols. It is, therefore, possible for Service Elements to generate data in an IPDR compliant format. Such an interface will be identified as an A interface with an IPDR compliant data format.

Typical requirements for the A interface (observed by the implementations of SE vendors, but not necessarily by IPDR NDM-U) are:

- Minimum encoding complexity
- Convenience of data collection and record construction
- Minimum data size
- Minimum imposition of data storage requirements on the SE
- Support for custom vendor features and capabilities

### 2.4.2.2 B Interface

The B interface delivers IPDRs from IPDR Recorders to IPDR Stores and to IPDR Transmitters. The payload conforms to the schema definition of an IPDR. As of this issue of the NDM-U, the transfer protocol for this interface is not specified.

### 2.4.2.3 C Interface

The C interface delivers IPDR documents from the IPDR Store to the IPDR Transmitter. Each IPDRDoc contains one or more IPDRs from a particular Service Element served by a given IPDR Recorder. As of this issue of the NDM-U, the transfer protocol for this interface is not specified.

### 2.4.2.4 D Interface

The D interface delivers IPDR documents from IPDR Transmitters to Business Support Systems. The transfer protocol on this interface is specified in Chapter 4.

### 2.4.2.5 E Interface

The E interface delivers IPDR documents from one Business Support Systems to another BSS. Various scenarios of providing service can be anticipated which will involve a combination of transport technologies and multiple service providers. In such scenarios, the business relationships involved will require the exchange of usage data for a variety of business process applications (e.g., net settlement, retail bill detail, customer service, fraud abatement, marketing studies). The interface dedicated to such exchange is the E interface. For IPDR-compliant service providers, the data exchanged may very well be IPDRDocs being delivered via some variant of the D interface protocol. In hybrid cases, the sending or receiving system may have to mediate the data format and protocol from or to an NDM-U-compliant form. Other industry standards and practices will require mappings with respect to NDM-U for this to be implemented. The definition of the details of the E interface will address this requirement.

Note that this document focuses on the definition of IPDR Documents, the information content of IPDR records and the interface between ITs and BSSs. The D interface is completely specified in this document while other interfaces are identified here to aid in decomposing the problem statement. Implementations adhering to this version of NDM-U are not required to explicitly separate the IPDR Recorder, IPDR Store and IPDR Transmitter roles, since the B and C interfaces are unspecified. Similarly, interfaces A and E are not explicitly addressed in this release of the NDMU. As of this writing, IPDR working groups are beginning to address interfaces A and E.

## 2.5 NDM-U Model Usage Scenarios

The focus for specifying requirements is given to D interface though there may be implied requirements to other interfaces. Section 3 provides general requirements and general usage attributes applicable to any IP service. Note that any new IP service added in the future may impact the general requirements and usage attributes. Section 0 provides IP services covered and yet to be covered in this chapter.

A number of assumptions are made in the following usage scenarios:

- Home Service Provider handles all business needs (via BSS applications) of the Service Consumer.
- Service Consumer may or may not be within the Home Service Provider's service area.
- BSS to BSS interfaces (E Interface) is outside the scope of this version of the NDM-U.

Some applications of NDM-U will result in large numbers of IPDRs being generated, requiring economical storage, transport, and processing implementations. Several requirements stated below are intended to address this assumption. However, no quantitative requirements regarding performance (end-to-end delay,

transfer rate, etc.) or efficiency (message size, compression ratio, etc.) will be stated in this document. The mechanisms designed in later chapters of this document, which satisfy the general requirements in this area, should give implementers adequate tools to make cost versus technology tradeoffs, justified in light of the business problem being solved. Product vendors designing implementations of this specification are assumed to be aware of the overall marketplace requirements for such systems and service providers selecting one or more of these implementations will be expected to require those vendors to demonstrate competitive features in the area of performance.

### 2.5.1 SE Directly Interfaced to BSS



Figure 5

A service element can directly interface with the business support system. If an SE node supports the D interface specifications including backing stores and the retransmission of IPDR documents, then the SE node can communicate directly with the BSS node. The IR, IS, and IT nodes do not necessarily have to be independent system entities. In this case the IR, IS, and IT nodes are incorporated within the SE node.

### 2.5.2 SE Interfaced to BSS Via Mediation System

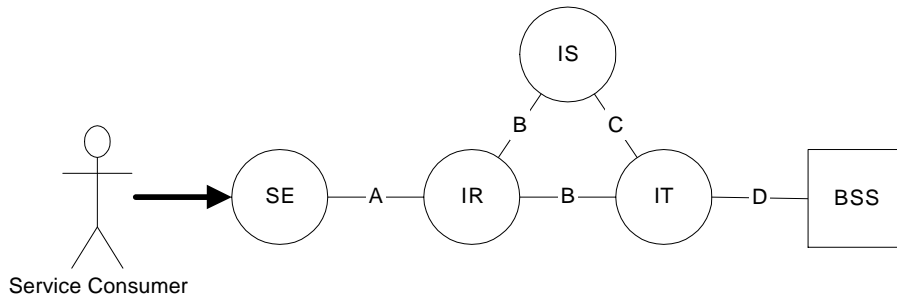


Figure 6

When an SE node provides a proprietary usage record interface or when it cannot provide record aggregation or it does not provide record retransmission capabilities, then an external IPDR compliant mediation system is required. The IPDR mediation system provides an interface to the SE node's proprietary protocol via the A interface, provides the required NDM-U D interface protocol, and it provides an IPDR backing store for re-transmitting the IPDR records to the BSS. This is the traditional IPDR NDM-U topology.

### 2.5.3 SE Interfaced to BSS Via Multiple Mediation Devices

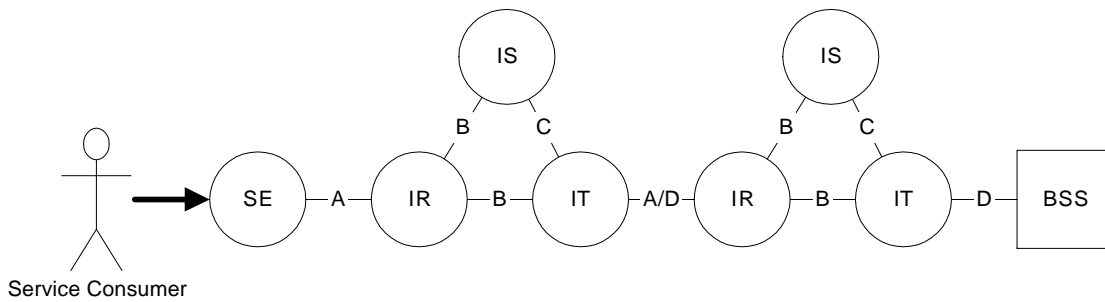


Figure 7

In some cases due to legacy mediation systems or for other reasons, more than one mediation system may be cascaded between the SE and BSS nodes. Two cases arise. In one case Mediation system 1 can transmit IPDRs to Mediation system 2 using the D interface protocol. In another case, if both mediation systems know a common proprietary protocol, compliant to the A interface specifications they can use that interface. In the former case, both mediation systems need to be IPDR compliant, while in the later case only the second mediation systems needs to be IPDR compliant. This illustrates the A interfaces requirement for being protocol compatible with the D interface.

### 2.5.4 Simple Roaming with Separate Access and Home Service Providers

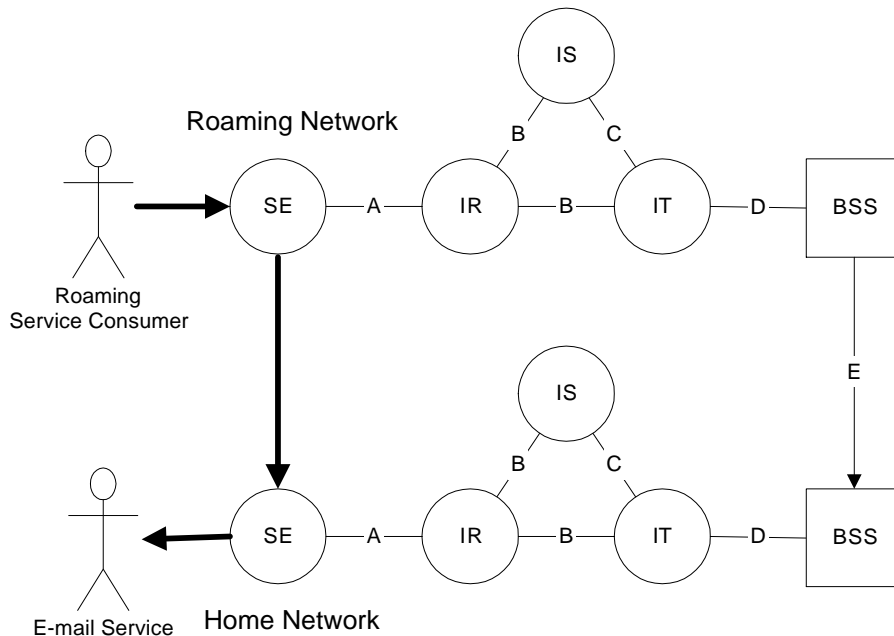


Figure 8

If an ISP offers roaming to its subscribers then the cooperating roaming ISP needs to be able to deliver its roaming IPDR settlement records to the home ISP BSS. This is implemented over the NDM-U defined E

interface. IPDR settlement records may be transmitted directly between participating BSS systems as illustrated here, or they may communicate via an intermediate clearing house BSS system. In either case, the settlement records are exchanged via the NDM-U E interface protocol standard.

### 2.5.5 Service Provider also Provides an Application Service

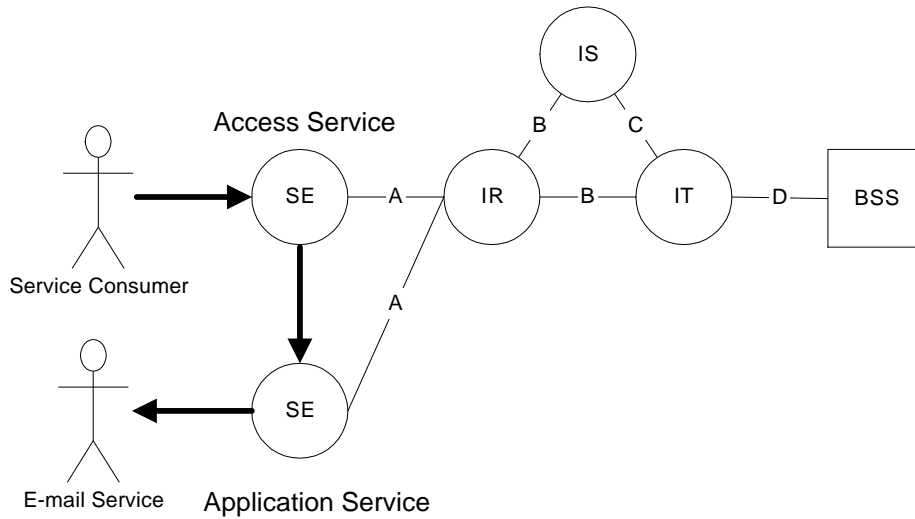


Figure 9

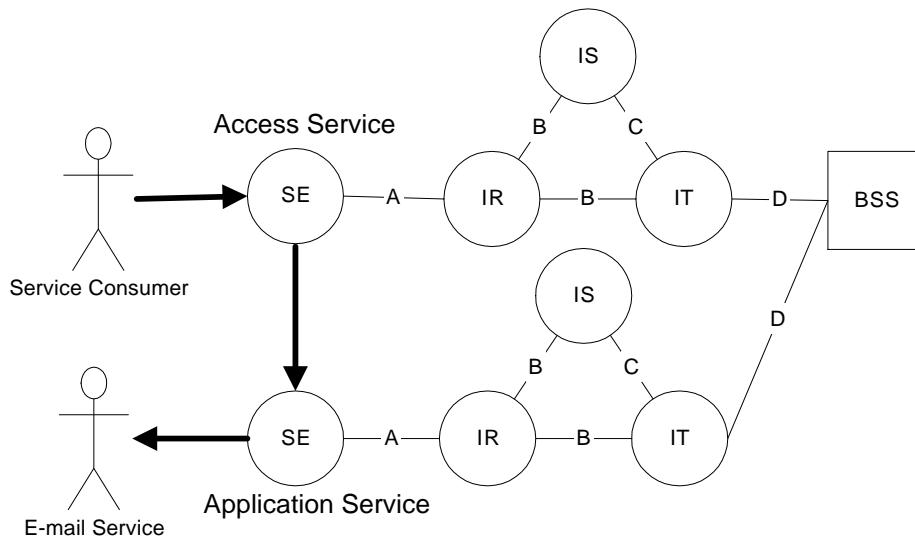


Figure 10

If an ISP wants to unbundle its own local IP services to its subscribers it may offer several application services on its own local network. Each of these services along with the basic access service would be treated as separate SE nodes feeding into the common BSS system. As in Figure 9 both SEs feed into a common IR node, which then make shared usage of the IS node and the IT node. Figure 10 illustrates where completely separate IR, IS, and IT node sets serve each SE but all the IPDR usage records are still transmitted to a common BSS system. IRs are capable of receiving IPDRs from multiple SEs.

### 2.5.6 Separate Home, Access, Transport, and Application Service Providers Acting in Concert

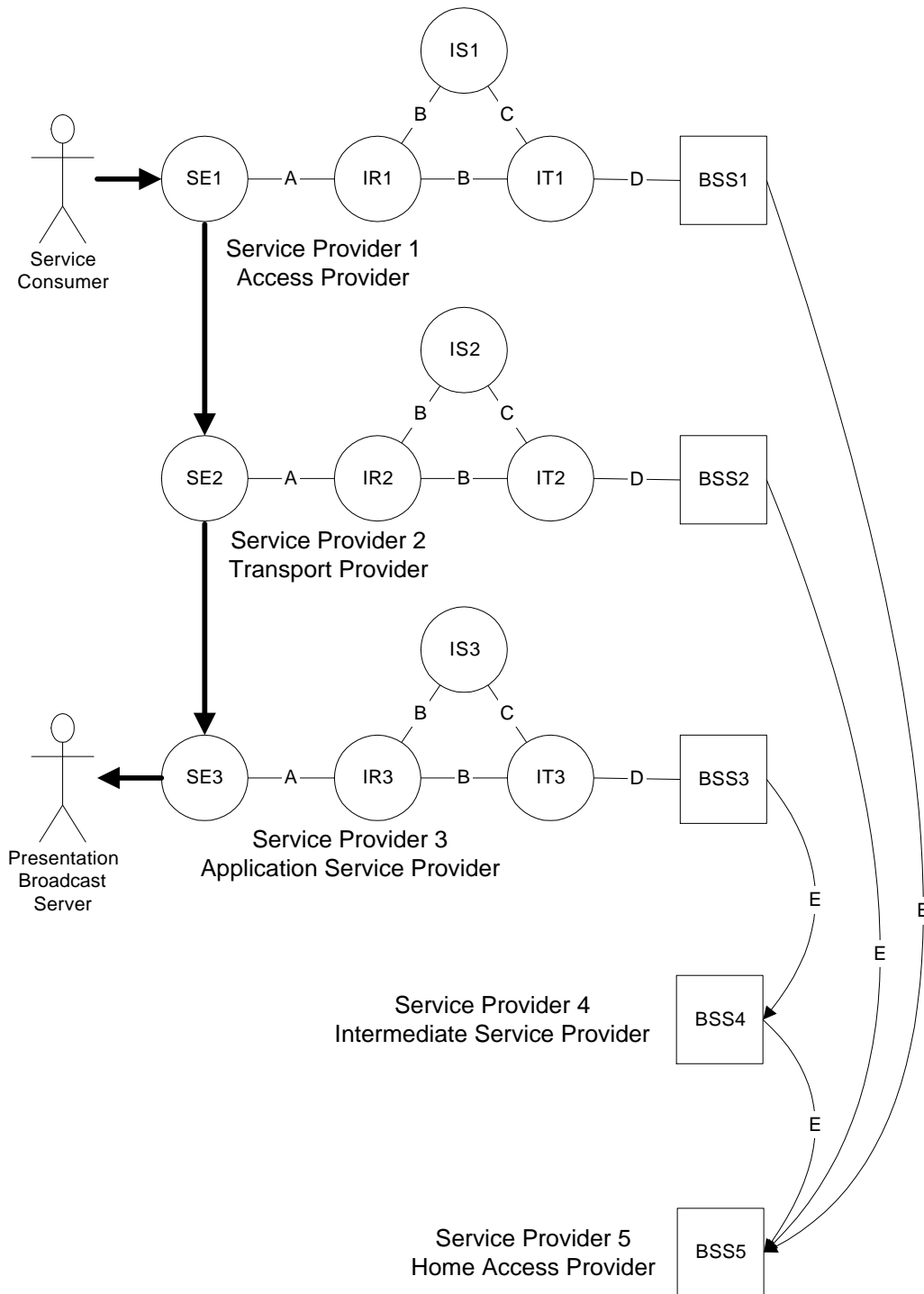


Figure 11

This scenario illustrates how IP service providers can jointly provide a realistic application to a service consumer. The service consumer (SC) is

1. Roaming outside of his home wireless service area
2. Using a digital cellular network interface (e.g. GPRS or CDMA2000) on a laptop computer
3. To participate in net meeting, using an internet connection to
4. View a broadcast Power Point presentation and
5. Call into a conference call bridge using VoIP;
6. The conference bridge is served by a PSTN, so the VoIP call is an IP-to-PSTN call, served by a gateway.

Figure 11 illustrates the interconnections among the service providers, while Figure 12 shows the sequence of IPDR message interactions between them. Five service providers participate in this scenario:

1. Access Service Provider (SP1) - this provider grants roaming access to the service consumer.
2. Transport Service Provider (SP2) - this provider interconnects service providers SP1 and SP3.
3. Application Service Provider (SP3) - this provider offers the application service, relying on one or more access/transport service providers to establish the session connection with service consumer.
4. Intermediate Service Provider (SP4) - this provider acts as an intermediate BSS on behalf of one or more other service providers. Such applications as service bureaus, clearing houses, rating bureaus, fraud bureaus, pre-paid authorization centers and other intermediate IPDR processing applications are examples of the role of this service provider.
5. Home Access Service Provider (SP5) – this is the service consumer’ s home account service provider, which ultimately bills the consumer. The other service providers deliver settlement IPDRs to SP5.

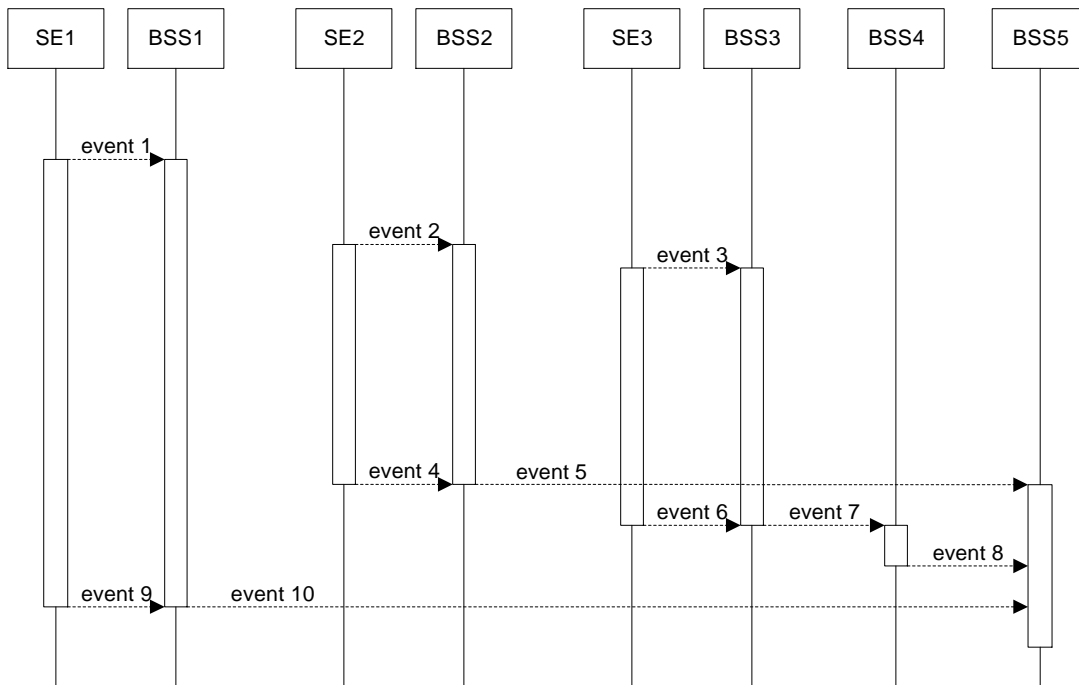


Figure 12

The following IPDR sequence of events occur:

1. SC initiates access request to the SP1. SE1 send start IPDR to BSS1
2. SC initiates VoIP call to conference bridge via SP2's gateway server. SE2 send start IPDR to BSS2.
3. SC initiates connection to SP3's presentation broadcast server. SE3 send start IPDR to BSS3
4. SC terminates VoIP call. SE2 send stop IPDR to BSS2
5. BSS2 send settlement IPDR to BSS5
6. SC terminates presentation session. SE3 send stop IPDR to BSS3.
7. BSS3 sends settlement IPDR to BSS4
8. BSS4 relays settlement IPDR to BSS5
9. SC terminates access. SE1 send stop IPDR to BSS1
10. BSS1 sends settlement IPDR to BSS5

To simplify this scenario the intervening IPDR messages passing to and from the IR, IS, and IT have been omitted in Figure 12. These IPDR messages have previously been discussed.

## 3. Business Requirements

### 3.1 Introduction

This chapter provides high-level requirements for BSS applications needs. It also provides the framework for specifying new IP services not already covered in the various standalone Service Specification documents.

Section 3.3 provides the general overview of the network model from previous chapters and its applicability to the BSS applications needs. The focus for specifying requirements is given to D interface though there may be implied requirements to other interfaces. Section **Error! Reference source not found.** provides general requirements and general usage attributes applicable to any IP service. Note that any new IP service added in the future may impact the general requirements and usage attributes. Section 0 provides IP services covered and yet to be covered in this chapter.

### 3.2 Assumptions

- Home Service Provider handles all business needs (via BSS applications) of the Service Consumer.
- Service Consumer may or may not be within the Home Service Provider's service area.
- BSS to BSS interfaces (E Interface) is outside the scope of this version of the NDM-U.
- Some applications of NDM-U will result in large numbers of IPDRs being generated, requiring economical storage, transport, and processing implementations. Several requirements stated below are intended to address this assumption. However, no quantitative requirements regarding performance (end-to-end delay, transfer rate, etc.) or efficiency (message size, compression ratio, etc.) will be stated in this document. The mechanisms designed in later chapters of this document, which satisfy the general requirements in this area, should give implementers adequate tools to make cost versus technology tradeoffs, justified in light of the business problem being solved. Product vendors designing implementations of this specification are assumed to be aware of the overall marketplace requirements for such systems and service providers selecting one or more of these implementations will be expected to require those vendors to demonstrate competitive features in the area of performance.

### 3.3 Generic Requirements

#### 3.3.1 Mediation

The general requirements for mediation (defined here as all those functions performed between the A and D interfaces) are, in almost all cases, service independent. Depending on the business model mediation tasks could span a wide variety of actions. However, in general terms mediation tasks include the collection, generation, aggregation and reconciliation of IPDRs across Service Elements, geographical areas and time.

1. Mediation shall support both polling and the producer can initiate pushing for data transfer, so that the data transfer either by the consumer or autonomously.
2. Mediation shall support data transfer for both individual events and batches of events.
3. Mediation shall support retrieval of IPDR documents.
4. Service elements shall be uniquely identified within the scope of each terminating IPDR Recorder.
5. Each IPDR shall have a unique event identifier within service elements. If IPDRs are related and the relation is visible to the IPDR Recorder (aggregator) then, a reference to the related record (base IPDR) shall contain this unique identifier.
6. IPDR shall enable the interim recording across multiple service elements and time. That is, enabling event information to exist in multiple records, over several IPDR documents.

7. Mediation shall support uniquely identifying IPDR documents for the purpose of gap and duplicate detection.

### 3.3.2 Format

This set of requirements pertains to the IPDR format.

1. The IPDR format shall be extensible permitting the addition of any set of services and service specific usage attributes.
2. The IPDR format shall be able to self-describe its usage attributes.
3. The IPDR format shall capture sufficient information to identify an IPDR Service Consumer.
4. The IPDR format shall provide specified data types, so that various systems can interpret the data properly. Times in IPDR should be expressed per ISO 8601 format for the purpose of facilitating data exchange. The specific time precision requirements vary with applications (e.g. IP packet time as opposed to billing time) and are individually specified in the attribute list. For billing purpose, time stamp accuracy should be 1 second or better. Local time zone offset with reference to GMT should be provided and should reflect local time of calling party for correct billing.
5. The IPDR format shall support efficient encoding.

### 3.3.3 Application Protocol

1. The NDM-U protocol shall support encryption of IPDR documents.
2. The NDM-U shall use open protocols and description languages.
3. NDM-U protocol/format shall separate the record format and exchange protocol.
4. NDM-U protocol shall support transfer capabilities negotiation.
5. NDM-U protocol shall support both individual and batch transfers of data
6. NDM-U protocol shall support resynchronization to a particular point in the order of delivery of IPDR documents.

### 3.3.4 Usage Attributes

1. The IPDR format specification shall indicate, for all usage attributes, if the information is required, optional or conditional.
2. The IPDR format specification shall indicate usage attributes data type.
3. Where appropriate, a data type of value/unit shall be specified to denote the unit of measure of an associated attribute value.

### 3.3.5 Settlement

1. The NDM-U protocol and format shall support roaming.
2. The NDM-U protocol and format shall support mobile service consumer.

## 3.4 Listing of Services

### 3.4.1 Services Covered

For describing the context environment of the business requirements listed in this chapter, a set of services is analyzed. Then, for each service multiple use cases are depicted. The list of services considered in this chapter is a representative and not a comprehensive list. This list will be augmented through contributions by other relevant standard bodies and through the progress of IPDR organization work. See references [7] through [14] for the definitions of these services.

Services considered by the BR working group in this version of the draft are:

1. Video on Demand (VoD)

Services considered by the BR working group in previous version of the draft are:

1. Application Services (ASP)
2. Voice over IP (VoIP)<sup>3</sup>
2. E-mail Services
3. Streaming Media
4. Authentication and Authorization Services (AA)
5. Internet Access (including wireless)
6. Content/Service (including wireless)
7. Push Delivery (including wireless)
3. Wholesale Requirements

### **3.4.2 Services for Future Consideration**

Since the list of services considered in this version is not a comprehensive list, and recognizing the importance of other services, we are including a list of services to be considered in future releases of this specification:

1. Virtual Private Networks (VPN)
2. Multi-party conferencing (video/voice)
3. E-commerce/M-commerce
4. Unified Messaging
5. Video conferencing over IP
6. IP television

### **3.4.3 Services Considered by other Organizations**

It is recognized that the specification of services requires expertise and experience in the providing or equipping such services. The IPDR organization encourages domain experts and service providers to submit specifications of services whose usage would be recorded by an IPDR Recorder. The form of such submissions should conform to the templates and guidelines described in this chapter.

---

<sup>3</sup> Considerable modifications have occurred to this specification.

---

## 4. NDM-U Protocol – IPDR Document Structure and Transfer

This section contains a complete discussion of protocol; namely, the notation, encoding, message format, procedure and semantics needed to implement the requirements specified in sections 2 and 3.

### 4.1 Overview of Encoding

This document defines two schemes for encoding IP usage records. One is a text-based XML encoding and the other is a compact encoding based on XDR. A one-to-one mapping exists between each encoding scheme. The XML encoding was developed first. Its strengths are that it is easy to read, debug, and deploy, while its weakness is that it has a low information density and can slow down the exchange of information. The compact encoding scheme was introduced in NDM-U v3.0 and is a high-performance alternative to XML, allowing information to be exchanged at faster rate due to the vastly higher information density of the usage records. XML Schema is used to completely describe the names, types and order of elements in the XML encoding. It also provides the starting point for the compact encoding since there is a one-to-one mapping between the XML encoding and the XDR-based compact encoding. Section 4.2 describes the XML encoding scheme and section 4.3 describes the compact encoding scheme.

### 4.2 XML Encoding

A single Master IPDR Schema Document declares elements common to all IP-based Services. A service-specific schema document is then used to define the usage attributes which describe usage events for a particular type of service. For example, an IPDR instance document that corresponds to Streaming Media services must account for the Movie Names, however an IPDR instance document corresponding to email services would not. For this reason Streaming Media and email services require separate service-specific schema documents.

The IPDR Document hierarchy allows an IPDRDoc to contain many usage records (IPDRs). Details about the consumer, service elements, and usage attributes are contained within each IPDR element. The master schema does not define any child elements for the IPDR element because these details are specific to a particular service and in fact may be different for each service. Note: Previous versions of the NDM-U defined child elements (SS), (SC), (SE), and (UE) for the IPDR element, these sub-elements have been deprecated and are no longer used in NDM-U 3.0 and subsequent versions.

Figure 4.1 graphically presents the different IPDR elements and their relationship to each other<sup>5</sup>.

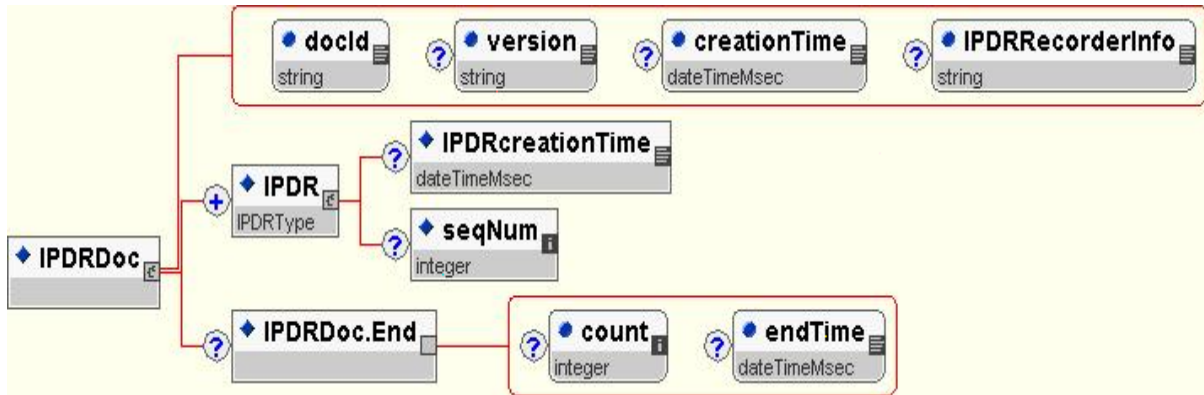


Figure 4.1

The document's main body is made up of one or more IPDRs that represent single usage events.

The document has an optional ending block of information represented by IPDRDoc.End.

An IPDR contains data describing the consumer, service provider, and metrics or parameters of a specific usage event. The usage represented may be measured as a discrete event, or part of an ongoing session.

Service specific schemas extend the base IPDR Schema by defining elements that are needed to characterize a usage event for that particular service.

### 4.2.1 IPDR Master Schema

The Master IPDR Schema Document formally describes how all IPDR documents are constructed. Additional details describing element use are included and considered part of the overall IPDR Document specification. Comments do not form part of the specification.

All service-specific schema documents include the Master IPDR Schema Document via the

```
<include schemaLocation="..." />
```

construct. Each IPDR instance document refers to one of the service-specific schema documents via the

```
xsi:schemaLocation="..."
```

attribute assigned to the <IPDRDoc> element.

The most recent version of the Master IPDR Schema Document is available at:

<http://www.ipdr.org/public/ipdrDoc3.0.xsd>

The entire Master Schema is presented below and is followed by an annotated section containing further description and restrictions on the various elements. The annotations are considered part of the specification. The service-specific schema and annotations for the actual service specifications for the services listed in Chapter 3 follow in References [7] through [13].

```
<?xml version = "1.0" encoding = "UTF-8"?>
<schema xmlns = "http://www.w3.org/2000/10/XMLSchema"
```

<sup>5</sup>The symbols in this figure denote the following: “?” – Optional, “+” – A set of one or more occurrences.

```

targetNamespace = "http://www.ipdr.org/namespaces/ipdr"
xmlns:ipdr = "http://www.ipdr.org/namespaces/ipdr"
version = "3.0">
<element name = "IPDRDoc">
  <annotation>
    <documentation>
      The IPDRDoc element is the top-level container of a set
      of IPDRs. The document will also define the entity
      which recorded these IPDRs via the IPDRRec element.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref = "ipdr:IPDR" maxOccurs = unbounded"/>
      <element ref = "ipdr:IPDRDoc.End" minOccurs = "0"/>
    </sequence>
    <attribute name = "docId" use = "required" type =
"string"/>
    <attribute name = "version" type = "string"/>
    <attribute name = "creationTime" type =
"ipdr:dateTimeMsec"/>
    <attribute name = "IPDRRecorderInfo" type = "string"/>
  </complexType>
</element>
<element name = "IPDRDoc.End">
  <annotation>
    <documentation>
      The IPDRDoc.End element optionally marks the end of the
      IPDR block. It may contain some check information like
      a count of IPDRs.
    </documentation>
  </annotation>
  <complexType>
    <attribute name = "count" type = "integer"/>
    <attribute name = "endTime" type = "ipdr:dateTimeMsec"/>
  </complexType>
</element>
<complexType name = "IPDRType"
  final = "restriction">
  <annotation>
    <documentation>
      This is the base type for the IPDR element. The
      service-specific schema can extend this by deriving
      from it.
    </documentation>
  </annotation>
  <sequence>
    <element name = "IPDRCreationTime" type =
"ipdr:dateTimeMsec" minOccurs = "0"/>
    <element name = "seqNum" type = "integer" minOccurs =
"0"/>
  </sequence>
</complexType>
<element name = "IPDR" type = "ipdr:IPDRType">
  <annotation>
    <documentation>

```

```

        An IPDR describes an event between a service
        consumer and a service element. Details of the event
        are contained within this record. All IPDR elements
        have a time indicating when the event occurred.
        </documentation>
    </annotation>
</element>
<simpleType name = "dateTimeMsec">
    <annotation>
        <documentation>
            This type supports time resolution at the millisecond
            level It is further constrained to always use the
            timezone designator "Z" indicating GMT. Quantities of
            this type can optionally use 3 digits of fraction after
            the second to represent the milliseconds. If absent, it
            is assumed the millisecond component is ".000".
            Example: 1999-05-31T13:20:00.561Z
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-
9]{2}:[0-9]{2}(\.[0-9]{3})?Z"/>
    </restriction>
</simpleType>
<simpleType name = "ipV4Addr">
    <annotation>
        <documentation>
            An IP version 4 address in dotted notation decimal. Example:
            15.13.120.22
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}"/>
    </restriction>
</simpleType>
<simpleType name = "ipV6Addr">
    <annotation>
        <documentation>
            An IP version 6 address in colon separated 2 byte block
            hexadecimal notation.
            Example: FEDC:AB19:12FE:0234:98EF:1178:8891:CAFF
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-
F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-
F]{4}:[0-9a-fA-F]{4}"/>
    </restriction>
</simpleType>
<simpleType name = "UUID">
    <annotation>
        <documentation>
            A universal unique id in hex dash notation.
            Example: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
        </documentation>
    </annotation>

```

```
<restriction base = "string">
  <pattern value = "[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
</restriction>
</simpleType>
</schema>
```

## 4.2.2 Annotated IPDR Master Schema

A description of each element in the IPDR Schema Document is presented below.

### IPDRDoc

```
<element name = "IPDRDoc">
  <annotation>
    <documentation>
      The IPDRDoc element is the top-level container of a
      set of IPDRs. The document will also define the
      entity which recorded these IPDRs via the IPDRRec
      element.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref = "ipdr:IPDR" maxOccurs =
unbounded"/>
      <element ref = "ipdr:IPDRDoc.End" minOccurs =
"0"/>
    </sequence>
    <attribute name = "docId" use = "required" type =
"string"/>
    <attribute name = "version" type = "string"/>
    <attribute name = "creationTime" type =
"ipdr:dateTimeMsec"/>
    <attribute name = "IPDRRecorderInfo" type = "string"/>
  </complexType>
</element>
```

The attributes of the IPDRDoc element are described below.

- docId – A Universally Unique Identifier (UUID) – *ed. note: this type is not defined as part of the base set of XML Schema datatypes. For the purposes of this document we will assume that it will be incorporated in a later draft (either XML Schema Datatypes or NDM-U)*
- version - identifies the version of the Master IPDRDoc Schema being used. This version shall be 30.
- creationTime - indicates the time this document was created. (See the “Generic Requirements” section for more information about timestamps”)
- IPDRRecorderInfo – identity of the IPDR Recorder responsible for the creation of this IPDRDoc.

### IPDRDoc.End

```
<element name="IPDRDoc.End">
  <complexType content="empty">
    <annotation>
      <documentation> The IPDRDoc.End element optionally marks the
      end of the IPDR block. It may contain some check
      information like a count of IPDRs.
    </documentation>
    </annotation>
    <attribute name="count" type="integer" use="optional"/>
    <attribute name="endTime" type="dateTimeMsec" use="optional"/>
  </complexType>
</element>
```

The attributes of the IPDRDoc.End element are described below.

- count - the number of IPDRs contained in this document (used as a check)..
- endTime - the time this document was completed. (See the “Generic Requirements” section for more information about timestamps”)

## IPDR

```
<element name = "IPDR" type = "ipdr:IPDRType">
  <annotation>
    <documentation>
      An IPDR describes an event between a service
      consumer and a service element. Details of the event
      are contained within this record. All IPDR elements
      have a time indicating when the event occurred.
    </documentation>
  </annotation>
</element>
<complexType name = "IPDRType"
  final = "restriction">
  <annotation>
    <documentation>
      This is the base type for the IPDR element. The
      service-specific schema can extend this by
      deriving from it.
    </documentation>
  </annotation>
  <sequence>
    <element name = "IPDRCreationTime" type =
"ipdr:dateTimeMsec" minOccurs = "0"/>
    <element name = "seqNum" type = "integer" minOccurs =
"0"/>
  </sequence>
</complexType>
```

The default child elements of the IPDR element are described below.

- IPDRCreationTime - the time the recorded usage event occurred. (See the “Generic Requirements” section for more information about timestamps”)
- seqNum - an optional integer value for auditing sets of IPDRs. The first IPDR in each IPDRDoc has a seqNum value of 0. Each subsequent IPDR within the same IPDRDoc has a monotonically increasing seqNum.

## IPDRtypes

```
<simpleType name = "dateTimeMsec">
  <annotation>
    <documentation>
      This type supports time resolution at the millisecond
      level It is further constrained to always use the
      timezone designator "Z" indicating GMT. Quantities of
      this type can optionally use 3 digits of fraction
      after the second to represent the milliseconds. If
      absent, it is assumed the millisecond component is
      ".000".
      Example: 1999-05-31T13:20:00.561Z
    </documentation>
  </annotation>
  <restriction base = "string">
```

```

        <pattern value = "[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-
9]{2}:[0-9]{2}:[0-9]{2}(\.[0-9]{3})?Z"/>
    </restriction>
</simpleType>
<simpleType name = "ipV4Addr">
    <annotation>
        <documentation>
            An IP version 4 address in dotted notation decimal. Example:
            15.13.120.22
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}"/>
    </restriction>
</simpleType>
<simpleType name = "ipV6Addr">
    <annotation>
        <documentation>
            An IP version 6 address in colon separated 2 byte block
            hexadecimal notation.
            Example: FEDC:AB19:12FE:0234:98EF:1178:8891:CAFF
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-
fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-9a-fA-F]{4}:[0-
9a-fA-F]{4}"/>
    </restriction>
</simpleType>
<simpleType name = "UUID">
    <annotation>
        <documentation>
            A universal unique id in hex dash notation.
            Example: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
        </documentation>
    </annotation>
    <restriction base = "string">
        <pattern value = "[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-
fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
    </restriction>
</simpleType>

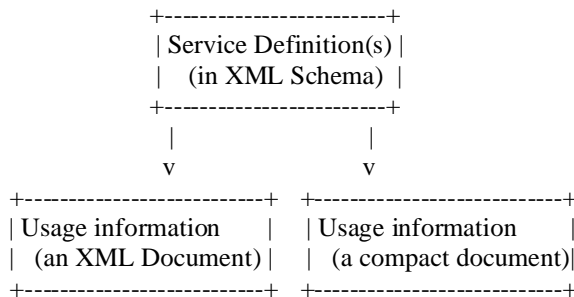
```

The schema fragment above is lexically included in the IPDRDoc schema to provide the default types required to be used in service definition schemas (see [7]).

## 4.3 Compact Encoding

### 4.3.1 Introduction to the Compact Encoding

This section defines a document format that offers a compact and efficient representation of usage accounting data. This format has the additional benefit of providing a well-defined equivalent XML encoding. Both the compact and XML formats are based on a common service definition specification defined by IPDR. The service specification is expressed as one or more XML Schema documents, which follow particular guidelines.



The details of the XML encoding are contained in Section 4.2.

The compact encoding represents integers and floating point values in a binary format as opposed to XML that uses an ASCII format. This increases the efficiency of reading and writing these values. The identification of usage attributes is done using the same ASCII names used in the XML format. These names are defined in the Service Definitions. XML namespace concepts that prevent name collisions are also supported.

The primary space savings comes from the separation of description of the usage attributes (i.e. their name and type) from the actual event values.

In situations where multiple instances of the same event format occur, this separation allows description information to be written only once in the document. Only the values are written for every event.

The exact format used to encode the values is defined in the RFC1832 External Data Representation (XDR).

The compact document format allows for both encoding and decoding entities to process the documents in a stream-oriented fashion. That is, a decoding entity does not need to read to the end of the document before beginning to extract information. Similarly, an encoding entity does not need to have all the information in memory before beginning to write the document.

This streaming property can be critical when exchanging large sets of accounting information.

### 4.3.2 Requirements

The IPDR Protocol Working Group set the following requirements for a compact encoding format:

- The format should be more space efficient than the XML equivalent (preferably by several times).

- The format should allow for efficient parsing and processing.
- Complexity should be minimized in order to spur adoption and inter-working systems, and aid in debugging.
- The format would ideally draw from existing standards work.
- The format should provide performance and size characteristics that are comparable to known vendor proprietary formats.
- There must be well defined mapping from the service definition to the encoding.
- The format should allow for easy extension and expansion of the of defined services.
- The format should be self-describing. That is, knowing an encoding is an IPDR document, a system should be able to extract the recorded usage information without additional external control.

#### 4.3.2.1 Space Efficient

The exact savings is dependent on the length of the attribute names from the service definition, the ratio of primitive types to string oriented data, and the number of occurrences of similar usage events in a document.

Consider a record which contains the following five usage attributes:

```
<IPDR xsi:type="AA-Type">
  <subscriberId>joe</subscriberId>
  <ipAddress>192.168.2.64</ipAddress>
  <nasIdentifier>nas1.foo.com</nasId>
  <acctInputOctets>13444</acctInputOctets>
  <acctOutputOctets>7777</acctOutputOctets>
</IPDR>
```

Not including blanks and linefeeds, this record occupies 219 bytes.

The compact encoding of this record is as follows:

```
00 00 00 02 00 00 00 01 FF FF FF FF 00 00 00 03
j o e 00 C0 98 40 20 00 00 00 0C n a s 1
. f o o . c o m 00 00 34 84 00 01 2F DF
```

The compact encoding takes 48 bytes and is about 4.5 times more compact than the equivalent XML encoding.

Note that one time header costs in either document format are not shown or counted. If only one or two usage events are encoded in an instance document, then the compact format savings will be significantly less than this factor. However as the number of events of a similar type increases, the savings approaches this value.

#### **4.3.2.2 Efficient parsing and processing**

Because numeric values are represented in binary format, they may be mapped directly into variables. Likewise when producing this format, variable contents may be simply written to the file.

The descriptor format allows processing entities to operate in a table driven approach to the production and consumption of different record types.

#### **4.3.2.3 Low Complexity**

There are only eight basic primitive types to encoded and decode. Record descriptors have a simple name based format. Records themselves consist of only simple primitive types. The overall document structure is closely related to XDR which allows the format to be described in a C-like language, which is concise and well understood.

#### **4.3.2.4 Existing standards work**

The encoding policy for primitive values is based on XDR [RFC1832], the service definition format is common with that for XML IPDR Services and is a subset of XML-Schema. Support for multi-byte character content is based on UTF-8 (RFC2279).

#### **4.3.2.5 Performance and size characteristics**

Preliminary results look promising. However this document can't provide any quantitative data since the member companies are somewhat reluctant to share this data.

#### **4.3.2.6 Well defined mapping**

The names of usage attributes are common in the XML and compact encoding model and derive from the same service specifications. [Note to ensure this, further restrictions on the use of XML-Schema were defined in NDM-U 3.0]

#### **4.3.2.7 Extension and expansion**

The means for extending service definitions by third parties as well as within IPDR.org has been clarified in NDM-U. It simply involves the creation of additional XML-Schema documents to describe the new service and the proper use of XML-Namespaces.

#### **4.3.2.8 Self-describing**

The document contains versioning information at the front of the header to support changes over time. The document contains references via URI to namespace and service definitions and contains descriptor blocks that identify usage attributes in events according to their XML-Schema based service definition. This produces a fully self-describing document.

Note that this assumes that the service definition URI information is present and accessible to the processing entity.

### 4.3.3 XDR Structure Definition

The compact encoding structure is defined using XDR notation. This provides a concise and readable means to convey the elements of the encoded format.

This definition first presents the XDR specification language for the format and then describes the key elements of the compact form in the following sub-sections.

```
/* IPDRDoc.xdr
 *
 * This file defines the structure of IPDRDocuments in a
 * compact format. It is expected that IPDRDocuments will
 * be stored in External Data Representation (XDR)
 * format according to the structures defined in this
 * file.
 *
 * XDR is defined in RFC1832.
 *
 * In order to accommodate efficient encoding of large
 * sets of data, the compact IPDR deviates from XDR
 * when encoding the length field associated with
 * IPDRRecordData and the set of IPDRStreamElements.
 *
 * In particular the length field uses a value of
 * 0xFFFFFFFF, in order to indicate that the element
 * count is indefinite. The number of elements for
 * these items is implicit as part of decoding the
 * content of those items.
 *
 * Note that only data structures are defined here.
 * This defines a format/structure to encode IPDRDocs it
 * does not define any transport.
 */

/*
 * Define our own string type, UTF8String. This emphasizes
 * that the content may not be ASCII, but may be UTF-8 encoded
 * representation of Unicode/ISO character set.
 *
 * Note that strings consisting of basic 7-bit ASCII characters
 * are unchanged when represented in UTF-8.
 */
typedef opaque UTF8String<>;

/*
 * Allowed primitive data types
 */
enum AttributeType {
    INTEGER          = 1,
    UNSIGNED_INTEGER = 2,
    HYPER            = 3,
    UNSIGNED_HYPER   = 4,
    FLOAT            = 5,
```

```
DOUBLE      = 6,
BYTE_ARRAY  = 7,
UTF8_ARRAY  = 8
};

/* AttributeDescriptor
 *
 * An attribute descriptor defines one attribute which will
 * appear in a record. It contains the attribute name and
 * the attributes type.
 */
struct AttributeDescriptor {

    UTF8String  attributeName; /* The name of an attribute. This name will either
                               * come from the default namespace or it will come
                               * from an alternate namespace as defined in the
                               * header. If an alternate namespace is used, the name
                               * will use XML syntax of the form nameSpaceId:attributeName
                               */

    Attribute  typeCode; /* Describes the type of this attribute. Explicitly
                          * including the type allows us to continue parsing
                          * a record even when the attribute name is unrecognized.
                          */
};

/* RecordDescriptor
 *
 * A record descriptor defines the set of attributes which are
 * carried in each record of a designated type. If different
 * subsets of attributes are carried in different records, then
 * a record descriptor is required for each.
 *
 * The record descriptor contains a sequence of attributeDescriptors
 * which identify the name and type of the attributes which are
 * associated with this record type. The type definition will
 * allow an entity which does not recognize some of the attributes
 * to still successfully parse the document.
 */
struct RecordDescriptor {

    int  descriptorId; /* The descriptorId is an integer which will allow
                       * individual records in a given document to specify
                       * which descriptor describes their layout.
                       */

    UTF8String  typeName; /* identifies a record type defined in a service definition.
                           * Record types specify the optional and mandatory fields.
                           * Note that there may be multiple descriptors associated
                           * with the same named type if they use different sets of
                           * optional attributes.
                           */
};
```

```
AttributeDescriptor attributes<>; /* Lists the attributes in the order they will appear
    * in records in this document. A record which points
    * to this descriptor's descriptorId, must exactly
    * match the set of attributes identified in this list.
    */

};

/* IPDRRecordData
 *
 * An individual record is a packed set of attribute values
 * determined by a recordDescriptor. The values are encoded
 * into the octet array according to their primitive format
 * and the XDR encoding rules. The overall length of the
 * opaque array is specified using the indefinite form (a
 * deviation from RFC 1832) in order to enable the output
 * to be produced efficiently.
 */
typedef opaque IPDRRecordData<>;

/* IPDRRecord
 *
 * An ipdrRecord structure represents a single recorded usage
 * event. It is a flat set of unique attributes. The specific
 * attributes are determined by the a recordDescriptor which is
 * referenced by index value.
 *
 * The set of attribute values are encoded according to CDR rules
 * as if a struct had been defined with elements of the type
 * identified in the recordDescriptor and in the order defined
 * by the recordDescriptor.
 */
struct IPDRRecord {
    int descriptorId; /* The index value of the descriptor which
        * describes this records attributes. It must
        * match the descriptor id in a recordDescriptor
        * encountered earlier.
        */

    IPDRRecordData data; /* The packed set of attributes */
};

/* IPDRDocEnd
 *
 * An IPDRDocEnd element is the last element in a sequence of
 * ipdrStreamElement's contained in the doc.
 */
struct IPDRDocEnd {
    int count; /* indicates the number of usage elements carried
        * in this stream of usage events, or -1 if
        * not used.
        */
};
```

```
hyper    endTime;    /* 64-bit time representation, defined to be
                    * the number of milliseconds since
                    * Jan. 1, 1970 0:00 GMT.
                    */
};

/*
 * Types of StreamElements
 */
enum ElementType {
    RECORDDESC = 1,    /* Describes a record structure */
    IPDRREC    = 2,    /* The values of a single record */
    DOCEND     = 3     /* Indicates the end of the element stream */
};

/* IPDRStreamElement
 *
 * An ipdrStreamElement defines the types of non-header
 * records which are contained in an IPDRDoc.
 */
union IPDRStreamElement switch (ElementType kind) {
    case RECORDDESC: RecordDescriptor desc;
    case IPDRREC:    IPDRRecord    rec;
    case DOCEND:    IPDRDocEnd    docEnd;
};

/* NamespaceInfo
 *
 * This associates a namespace URI with a simple identifier. It is used
 * to incorporate namespaces besides default from which attributes
 * are defined.
 */
struct NamespaceInfo {

    UTF8String namespaceURI; /* The URI for an alternate namespace. For example
                            * http://www.foo.com/ipdr/namespace
                            */

    UTF8String namespaceID; /* A string identifier which will be used to prefix attributes
                            * which come from this namespace (e.g. "foo" would
                            * result in attributes being named like "foo:specialAttr"
                            */
};

/* IPDRHeader
 *
 * The IPDRHeader contains general information about this document,
 * including versioning, information about the recorder and
 * references to IPDR Service definitions and their namespaces.
 */
```

```

struct IPDRHeader {

    int        version;        /* The compact format version.  Currently = 1 */

    UTF8String ipdrRecorderInfo; /* Identification information for the producer
        * of this document.  Typically URI, or blank
        */

    hyper      startTime;      /* Number of seconds since the Epoch
        * (00:00:00 UTC, January 1, 1970)
        */

    UTF8String defaultNamespaceURI; /* Identifies the default Namespace associated with
        * this record.  Those attribute names which are
        * unqualified will be assumed to be from this namespace
        * (e.g. "http://www.ipdr.org/namespace")
        */

    NameSpaceInfo otherNameSpaces<>; /* Identifies additional namespaces from which
        * attributes are derived.  This list may be empty.
        */

    UTF8String serviceDefinitionURIs<>; /* Identifies the set of service definitions from
        * which the records are produced.
        */

    opaque docId<>;            /* The UUID associated with this document */

};

```

```

/* IPDRDoc
 *
 * An IPDRDoc represents a collection of usage events recorded
 * in a compact binary format consistent with the External
 * Data Representation (XDR defined in RFC1832).
 *
 * The information content of the compact format is equivalent
 * to that contained in an XML encoded form.
 *
 * Note that the length of the elements array uses the
 * indefinite form (a deviation from RFC1832).  To support
 * efficient encoding of large sets of events.
 */

```

```

struct IPDRDoc {

    IPDRHeader    header;      /* contains information describing this
        * IPDR document
        */

    IPDRStreamElement elements<>; /* The descriptors and ipdrRecords may be
        * intermixed.  An ipdrRecord may not refer
        * to a descriptor which has not previously
        * appeared.  The ipdrDocEnd must be the final

```

```

        * element.
        */
};

```

### 4.3.4 Compact Format Definition

The basic structure is shown below:

```

+-----+
| Header |
+-----+
| Descriptor |
+-----+
| Usage Event |
+-----+
...
+-----+
| Usage Event |
+-----+
| Doc End |
+-----+

```

Multiple descriptors may appear in a single document. The descriptors may be intermixed with usage events. The only requirement is that a descriptor must appear before any usage event which references it.

The Doc End element may appear exactly once and must be the last element in the document.

It should be noted that in order to support the stream based encoding and decoding there are two points where arrays will be encoded using an indefinite length indicator. This technique is a deviation from RFC1832, but is necessary in order to reduce the possible memory and data copy requirements which could result from requiring the length be calculated up front.

The use of indefinite length does not produce any ambiguity when decoding the document, because in both cases where it is used, the process of decoding the indefinite length content will clearly identify where the end of the indefinite length set is located.

#### 4.3.4.1 Compact Document Header

The document header is structured as follows.

```

+-----+
| Version |
+-----+
| Recorder Info |
+-----+
| Creation Time |
+-----+
| Default Namespace |
+-----+
| Other Namespaces |
+-----+
| Service Defs |
+-----+

```

---

| Document UUID |  
+-----+

#### 4.3.4.1.1 Version

The version number is recorded as a 32-bit integer.

The version number associated with this specification shall be 1.

#### 4.3.4.1.2 Recorder Info

The recorder info is encoded as a 32-bit length indicator followed by a sequence of bytes of the specified length. Padding with bytes containing zero are used to ensure the set of bytes ends on a 4-byte boundary (in accordance with XDR encoding policy).

The length specifies the number of bytes (not including any padding). The length does not include the 4 bytes which the length indicator itself occupies.

The content of the Recorder Info is not specified in this document. It may be a URI reference or other information which aids in the identification of the element which created the document.

#### 4.3.4.1.3 Creation Time

The creation time is a 64-bit integer value representing the time the recording entity began creating this document. The value represents the number of milliseconds since Jan. 1, 1970 0:00 GMT.

#### 4.3.4.1.4 Default Namespace

The Default Namespace is encoded as a 32-bit length indicator followed by a sequence of bytes of the specified length. The Namespace, if present, should be in the form of a URI.

This namespace creates an association between unqualified attribute names and the default namespace, as specified in the "Namespaces in XML" W3C Recommendation.

#### 4.3.4.1.5 Other Namespaces

The Other Namespaces item is encoded as a 32-bit integer representing the number of namespace items, followed by that many namespace item.

Each namespace item represents a pair of UTF-8 strings which contain a prefix and a namespace URI. Each of these UTF-8 strings is encoded as a 32-bit length indicator followed by a sequence of bytes of the specified length.

If present these namespace items will be used to associate qualified attribute names with a given namespace as specified in the "Namespaces in XML" W3C Recommendation.

These namespaces in turn are associated with Service Definitions as follows:

The array of Namespace items must be equal to or less than the number of Service definition items. A Namespace item in the array of Other Namespaces is associated with the service definition in the same position in the array. Any service definition which does not have a corresponding Namespace item is considered to be associated with the default namespace.

A namespace item may have a zero length prefix name and namespace URI component. In the namespace item is considered to represent the default namespace.

See "XML Schema - 1 Structures" for a discussion of the association of Namespaces and Schemas.

#### **4.3.4.1.6 Service Definitions**

The Service definitions are encoded as 32-bit integer representing the number of UTF-8 strings which represent the service definition URIs. It is recommended that these URI's be accessible to the reader.

Each Service definition is a 32-bit length indicator followed by a sequence of bytes of the specified length.

It is assumed that the service definition URL's point to instances of XML-Schema documents which are further constrained to conform to the guidelines defined by the IPDR Service Definition Guide.

These Service Definitions will contain XML element definitions including typing and descriptive information. This information can be used to by a consuming entity to derive additional details about the individual usage attributes.

The usage attributes referenced in the document descriptors should correspond to entity definitions (with the appropriate namespace qualification) appearing in these Service definitions.

Note however if attributes do not appear in a Service definition or no service definitions are present or the service definition URIs are unreachable, the document itself is still able to be processed. There will merely be less explicitly available information about the attributes. And there is a greater risk that documents may refer to attributes using the same name with a different meaning.

#### **4.3.4.1.7 Document UUID**

Each document produced by a recorder will have a globally unique identifier to aid in the tracking of usage information delivery.

The UUID shall be encoded as a 32-bit length specifier (which will always have the value 16) followed by the 16-byte UUID sequence. UUID's are defined in the OSF specification for Distributed Computing Environment (DCE).

#### **4.3.4.2 Stream Elements**

The compact document format contains a header followed by a sequence of Stream Elements. Stream Elements are one of three types:

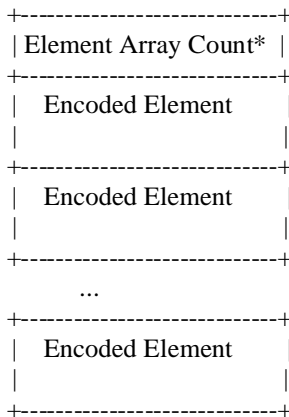
- Descriptor - a descriptor defines the structure of an IPDR Record. The descriptor is composed of a sequence of Attribute Descriptors which define the name and type of each Usage attribute as it will appear in an record.

- IPDR Record - a record consists of a reference to a descriptor followed by a sequence of values encoded in the order defined by the descriptor. Each value is encoded following the rules for the primitive type identified in the descriptor.
- Document End - this structure is used to mark the end of a series of stream elements. It also contains a count of IPDR Records and a timestamp indicating when the recorder completed production of this document.

In the XDR description language presented earlier the sequence of Stream elements is defined in the IPDRDoc structure as:

```
IPDRStreamElement  elements<>;
```

XDR specifies the rules for encoding the variable length array of elements as encoding a 32-bit integer indicating the total number of elements followed by the encoded instances of those elements.



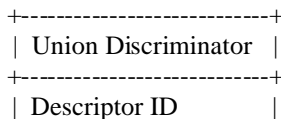
Requiring that the number of elements be encoded before the elements themselves are laid down in the record is impractical in applications where a very large set of usage events are to be recorded. In some cases it is desirable to begin recording the set of events before all the usage events have even arrived. In this case the recording entity will not have knowledge of how many elements are to be produced at the time the array length is to be encoded. This specification deviates from the XDR encoding, by instead requiring that the length always be specified by the value 0xFFFFFFFF. This indicates that the number of elements must be determined through the process of decoding the sequence of elements.

The end of the sequence of elements is unambiguously determined by the presence of a Document End element. After decoding this element the sequence is considered complete.

#### 4.3.4.2.1 Record Descriptor

The record descriptor defines the structure of one set of usage events which may appear later in this sequence of stream elements.

It is encoded as follows:



```

+-----+
| TypeName      |
+-----+
| Attribute Desc Count |
+-----+
| Attribute Descriptor |
+-----+
| Attribute Descriptor |
+-----+
...
+-----+
| Attribute Descriptor |
+-----+

```

The union discriminator field is the mechanism used by XDR to distinguish among different instances of a union type. This field is always represented by a 32-bit integer. Stream elements which are Record Descriptors use the value 1, to distinguish them from other stream element types.

The descriptor ID is a 32-bit integer identifier used to distinguish between other record types which may appear in the same stream. It is recommended to begin issue descriptor ID's beginning with 1 or 0 and assign sequential id's as new descriptors are added to a given document.

The typeName is a 32-bit length indicator followed by a sequence of bytes of the specified length. It is intended to refer to a ComplexType definition which appears in a service definition. The typeName may be namespace qualified if the type definition is from one of the other namespaces and not the default namespace.

The attribute descriptor count is encoded as a 32-bit integer. It represents the number of attribute descriptors which are present.

The order of the attribute descriptors indicates the order in which values for this record type will be encoded in IPDR Record data stream elements.

Each attribute descriptor consists of two fields:

```

+-----+
| Attribute Name |
+-----+
| Primitive Type |
+-----+

```

#### 4.3.4.2.1.1 Attribute Name

The Attribute Name is encoded as a 32-bit length indicator followed by a sequence of bytes of the specified length.

The attribute name is a character string identifying an attribute. The name is formulated according to the conventions defined in the W3C Recommendation, Namespaces in XML. Specifically the presence of the ":" character is used to indicate a non-default namespace. The substring on the left of the ":" should match a prefix defined in the Other Namespaces section of the header. If no ":" is present then the name is considered to be taken from the default namespace. In either case, the name should match an attribute definition in one of the referenced service definitions. Although the presence of the service definitions are not required, if there is no match, then additional information about the attribute will not be directly known to a consuming application.

An application may have additional information about a named attribute through external means not provided in this specification. However in this situation it should be noted that the self-describing nature of the document is diminished.

#### 4.3.4.2.1.2 Primitive Types

The primitive type is encoded as a 32-bit integer value.

The atomic primitives provided by this specification are the following, have the following identifier assignments:

```
INTEGER      = 1,  
UNSIGNED_INTEGER = 2,  
HYPER       = 3,  
UNSIGNED_HYPER = 4,  
FLOAT       = 5,  
DOUBLE      = 6,  
BYTE_ARRAY  = 7,  
UTF8_ARRAY  = 8
```

An Integer is a 32-bit signed integer value.

An Unsigned integer is a 32-bit unsigned integer value

A Hyper is a 64-bit signed integer value.

An Unsigned hyper is a 64-bit unsigned integer value.

A Float is a 32-bit floating point value.

A Double is a 64-bit floating point value.

A Byte array is a sequence of bytes of a specified length.

A UTF8 array is a byte sequence which represents the UTF8 encoding of a string of characters. The length indicates the number of bytes which UTF8 encode this string. Strings are not null terminated. ASCII strings which are made up of characters exclusively in the range 0x00-0x7F do not require any transformation to be represented in UTF8. Character code points which are in the range 0x80-0xFF and those which require multiple bytes have a simple mapping function defined in RFC 2279.

UTF8 is defined as a separate primitive type, because the use of information encoded as strings is a broadly applicable distinction from data which consists of a sequence of byte values which do not represent character code points.

#### 4.3.4.2.1.3 First Normal Form

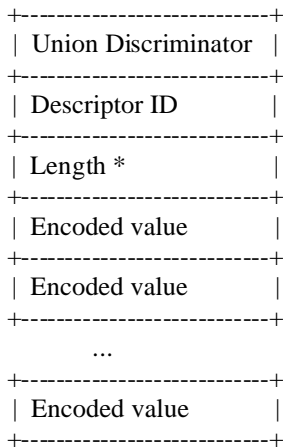
The sets of information which can be encoded for a single usage event are considered to be in "first normal form". First normal form is a basic relational database concept which states that the basic properties of tables are that they are made of atomic primitives and do not have multiple values for any attribute.

Usage events which consist of lists of information can still be represented by dividing the information content into different record descriptors and linking the events with some common identifier(s).

Alternatively some simple lists could be carried in string or byte array fields, although this is discouraged, because the self-describing nature of the document, in particular the embedded list, is diminished.

#### 4.3.4.2.2 IPDR Record Data

The record data identifies the descriptor that defines the layout of the encoded values followed by the encoded values themselves packed according to the XDR encoding rules for the primitive type associated with each attribute.



The union discriminator field is the mechanism used by XDR to distinguish among different instances of a union type. This field is always represented by a 32-bit integer. Stream elements that are IPDR Record Data use the value 2, to distinguish them from other stream element types.

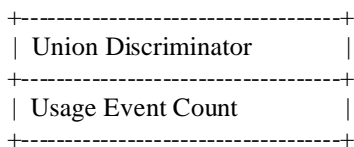
The descriptor id is a 32-bit integer identifier that should match a descriptor id defined previously in this document instance in a Record Descriptor stream element.

The encoding of the length element diverges from the XDR specification. Ordinarily this would represent the number of bytes that make up the series of encoded values contained in the data. However, because the encoded values may contain variable length byte arrays, and strings that must be UTF-8 encoded, it can be inefficient to calculate the length separately from the process of encoding the values. Therefore the length is represented by the value 0xFFFFFFFF that indicates that the individual values must be decoded according to the specified record descriptor.

The end of the sequence is determined unambiguously upon decoding the last value of the sequence of attributes.

#### 4.3.4.2.3 Document End

The document end identifies the last element in the sequence of stream elements. It also provides summary information about the document. Specifically the number of usage events in the stream and the time when the document was completed are recorded.



| Document Completion Time |  
+-----+

The union discriminator field is the mechanism used by XDR to distinguish among different instances of a union type. This field is always represented by a 32-bit integer. The Document End stream element uses the value 3, to distinguish it from other stream element types.

The usage event count is encoded as a 32-bit integer value. It should match the number of IPDR Record Data stream elements that appeared in this document.

The completion time is encoded as a 64-bit integer value representing the number of milliseconds since Jan. 1, 1970 0:00 GMT.

### 4.3.5 References

Note that all RFCs are available from <http://www.ietf.org/rfc/rfcNNNN.txt>.

All W3C recommendations are available from <http://www.w3.org/TR/#Recommendations>.

Publicly available IPDR specifications are available from:  
<http://www.ipdr.org/technical-work/document-releases.htm>.

R. Srinivasan, "XDR: External Data Representation Standard",  
RFC 1832, August 1995

Bray, T., J. Paoli, and C. Sperberg-McQueen, "Extensible  
Markup Language (XML) 1.0", W3C Recommendation, 2nd edition 6 October 2000

"Namespaces in XML", W3C Recommendation, January 1999.

"XML Schema Part 2: Datatypes", W3C Recommendation, May 2001.

"XML Schema Part 1: Structures", W3C Recommendation, May 2001.

IPDR Service Definition Guidelines.

T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, "Uniform  
Resource Identifiers (URI): Generic Syntax". RFC 2396,  
August 1998.

N. Brownlee, A. Blount, "Accounting Attributes and Record Formats",  
RFC 2924, September 2000

K. McCloghrie, J. Heinanen, W. Greene, A. Prasad, "Accounting  
Information for ATM Networks", RFC 2512, February 1999

F. Yergeau, "UTF-8, a transformation format of ISO 10646",  
RFC 2279, January 1998.

OSF specification for Distributed Computing Environment (DCE).  
OSF CAE Specification, Document C706, 1997

P. Leach, R. Salz, UUIDs and GUIDs, IETF draft, 1998. Expired,  
but available at: <http://www.webdav.org/specs/draft-leach-uuids-guids-01.txt>

---

## 4.4 Document Transfer

### 4.4.1 Introduction

First, it should be noted that the transfer protocol described in this version of the document is limited to the D interface.

The NDM-U Transfer Protocol provides three models of delivery:

- IT Push, where the IPDR Transmitter is responsible for delivering IPDR documents to a known BSS system.
- BSS Pull, where a BSS system requests specific documents from an IT.
- Demand Poll, where the IT system notifies the BSS of the availability of IPDR documents and the BSS is responsible for pulling them.

The Transfer Protocol is defined in a transport neutral manner. The working groups of IPDR.org will define specific mappings. In 3.0 there is a single specified transport mapping, the file based transfer protocol. This file based mapping is a minimal implementation of the demand poll model. For the file based mapping, a subset of capabilities are supported and communication is done through the file system rather than over the network.

A preliminary SOAP mapping was specified in NDM-U versions 2.0, 2.5 and 3.0. It has been removed from version 3.0. This decision was based on the lack of implementation and operational experience. A full mapping of the operations defined in this section are an area for further study.

The following sections define the Protocol:

- “Groups, Sequence numbers, Document ids and Subscriptions” describes how the IPDR Transmitter organizes IPDRDocs for retrieval and delivery to BSS systems. The protocol primitives are dependent on these concepts.
- “NDM -U Protocol Primitives and Parameters” specifies the set of operations available in the 3.0 version of the Transfer Protocol, and illustrated by the State Diagrams. These are again presented in a protocol neutral format. One exception is the definition of a “Capability file”. This XML document is further described in the next section.
- A state diagram view of the protocol interchanges, note that this does not specify any particular encoding or transfer mechanism. The protocol neutral model allows for the assignment of different mappings for the same basic protocol..
- “Capability Files, A Means for Subsequent Extension” describes how the XML based capability record can be exchanged between an IT and BSS, and how extensions are managed.
- “Simple File Scheme Mapping,” describes the file based transfer protocol used in several IPDR interoperability demonstrations. Note that this is now considered an instance of the defined protocol (albeit a trivial one).
- “Security Considerations”, describes the security mechanisms available with the currently defined protocol mappings.

## 4.4.2 Groups, Sequence Numbers, Document Ids and Subscriptions

An IPDR Transmitter organizes IPDRDocs into groups. A sequence number and a unique document id identify the documents in a group. BSS systems access IPDRDocs by explicitly pulling specific documents, or by subscribing to a group to request delivery by the IPDR Transmitter.

### 4.4.2.1 Groups

An IPDR Transmitter will receive either IPDR Documents from an IPDR Store or individual IPDRs from one or more IPDR Recorders. If individual records are received, it is the responsibility of the IPDR Transmitter to assign sets of records to an IPDRDoc and to place them in a logical IPDR Store.

Each IPDR Document the IPDR Transmitter must process is assigned to zero or more groups. The business rules, which assign IPDR Documents to groups, are a local matter for the IPDR Transmitter. Similarly the set of groups maintained by the IPDR Transmitter is a local matter.

Groups are named with a string identifier. The assignment of group names are a local matter, the only restriction is that an IPDR Transmitter with multiple groups must have a different name for each group. Two different IPDR Transmitters may use the same group name, but no relationship between these groups is implied.

### 4.4.2.2 Group Sequence Numbers

When the IPDR Transmitter assigns an IPDR Document to a group the document is given the next available sequence number for that group. Sequence numbers within a group are never repeated. To guarantee this property 64-bit integers may be needed to prevent roll over.

Sequence numbers are positive integers that increase by one for each new document. When a new group is created, the first document assigned to that group should be given the sequence number of 1.

As IPDR Documents are aged off based on policy, the corresponding sequence number within that group is no longer considered available.

### 4.4.2.3 Subscriptions

The NDM-U Transfer protocol provides a Push, Pull and Demand Poll based model for delivering IPDR Documents.

The concept of a subscription is introduced for the Push and Demand Poll models. A BSS system subscribes to a document group maintained by the IPDR Transmitter. During the life of this subscription the IPDR Transmitter will attempt to transfer each document in that group to the BSS. As new documents are added to the group these will also be delivered. Note that in the case of Demand Poll the document identifiers are delivered, but the BSS must pull the IPDRDoc itself.

Subscriptions may be explicitly requested by a BSS system through the subscribe primitive. Subscriptions may also be explicitly ended by a BSS system using the unsubscribe primitive.

Alternatively subscriptions may be set up on the IPDR Transmitter through local means (i.e. configuration).

During the subscription, the IPDR Transmitter will attempt to deliver each IPDR Document in the associated group to the configured BSS system exactly once.

The simplest mechanism to accomplish such a transfer is to serially deliver each document in order of their assigned sequence numbers.

Alternative mechanisms which deliver documents in parallel over multiple communication channels to the BSS may also be desirable, however the mechanism to recover documents is not defined by this protocol.

#### **4.4.2.4 Document Ids**

An IPDR Doc should have a globally unique identifier associated with it. These are Universal Unique Identifiers (UUIDs).

This identifier is assigned when the IPDR Document is created. When the IPDR Transmitter assigns a document to a group, the document's Id is not modified. Rather the document is assigned a sequence number within that group. If the same document is assigned to multiple groups, the sequence number assigned for each group is unrelated. However the document maintains its single unique document identifier, so that instances of the same document in multiple groups can be identified.

The definition of Universal Unique Identifiers resides in the OSF specification for Distributed Computing Environment (DCE). OSF CAE Specification, Document C706, 1997, Appendix A, located at:

<http://www.opengroup.org/onlinepubs/009629399/>

UUIDs are equivalent to Microsoft's Globally Unique Identifiers (GUIDs).

An open source C implementation of UUID generation is available in the appendix of the IETF draft, draft-leach-uuids-guids-01.txt. This draft has expired, but an archived copy is available at:

<http://www.ipdr.org/public/draft-leach-uuids-guids-01.txt>

Note: the IETF draft was allowed to expire because the group considered the OSF work a referenceable standard and did not chose to duplicate it.

### 4.4.3 Protocol Primitives and Parameters

The following sections summarize the parameters to each NDM-U Transfer protocol primitive.

Unless otherwise indicated all parameters are required.

#### 4.4.3.1 CapabilityReq

A request for a BSS to determine the modes of transfer and the versions of the transfer protocol supported by an IT.

It has the following parameters:

- version - the version of the NDM-U protocol in use. (string, e.g. "3.0"). Another level of subidentifiers could be introduced to represent bug fixes or minor corrections (e.g. 3.0.1), but is discouraged. This version uses the major revision number 3 and minor number of 0.
- requestorId - an identifier of the requestor. This is useful for log and audit purposes today and BASIC authorization. (string, URL identifying the requestor, e.g. "http://voip\_bss.example.com:6615/bss1").

#### 4.4.3.2 CapabilityRsp+

The response carries a description of all available NDM-U transfer capabilities supported by this IPDR Transmitter (IT).

NOTE: The XML encoding of the capabilities is considered the canonical description for all subsequent IPDR Transmitter protocol revisions and alternate mappings. All IT implementations must support exporting information regarding their capabilities via flat files in this format. Similarly it is assumed that systems interested in accessing the services of an IT implementation will be able to consume this description.

It has the following parameter:

- supportedProtocolList - a list of items, which describe the capabilities of this transmitter. Each item consists of three subfields, plus optional extensions:
  - version - a protocol version that this IPDR Transmitter supports. (string, e.g. "3.0")
  - primitiveList - a list of the operations. A single name identifies the request, response+ and response- primitives, which are supported by this IPDR Transmitter. (string list, e.g. "Capability, ListGroups, Subscribe, and Push"). Note that Unsubscribe is implied if Subscribe is offered.
  - protocolMapping - a specific transport that is supported. Currently the only mapping specified is "File".
  - extension - a container for version specific information. The contents of the extension are determined by the version and protocolMapping, and can be ignored by systems not supporting them. For version 3.0 the only supported subfield is identification information for the transmitter.

Example:

```
<extension>
<transmitterId>http://voip_it1.example.com:6614/voip_it</transmitterId>
</extension>
```

See the following section on Capability Files and Extensions for a complete example.

#### 4.4.3.3 CapabilityRsp- (see general NegativeRsp)

Rejects a capability request.

#### 4.4.3.4 ListGroupsReq

A request for the BSS to determine the available groups of an IT.

It has the following parameters:

- version - the version of the NDM-U protocol in use. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. This is useful for log and audit purposes today and BASIC authorization. (string, URL identifying the requestor, e.g. " http://voip\_bss.example.com:6615/bss1")

#### 4.4.3.5 ListGroupsRsp+

Carries the response to the request.

It has the following parameters:

- groupInfoList - a list of items, which describe the groups currently available from this transmitter. Each item consists of five subfields:
  - groupId - the name of a group supported by this transmitter. (string, e.g. " voip1\_group")
  - beginTime - the time the first document currently available from this group was created. (timestamp, e.g. "2000-12-31T23:59:00Z")
  - beginSeqNum - the sequence number of this document within the group. (integer, e.g. 7013)
  - endTime - the time the last document currently available from this group was created. (timestamp, e.g. "2001-01-10T23:59:00Z")
  - endSeqNum - the sequence number of this document within the group. (integer, e.g. 10987)

#### 4.4.3.6 ListGroupsRsp- (see general NegativeRsp)

Rejects a group listing request.

#### 4.4.3.7 ListDocsReq

A request for the BSS to determine the documents in a group.

It has the following parameters:

- version - the version of the NDM-U protocol in use. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. This is useful for log and audit purposes today and BASIC authorization. (string, URL identifying the requestor, e.g. " http://voip\_bss.example.com:6615/bss1")
- groupId - the name of the group on this transmitter to have its documents listed. (string, e.g. "voip1\_group")
- [optional any one of the following three, if none are specified it is equivalent to sinceSeqNum=0]
  - sinceTime [optional] - only list documents in this group, which were since the specified time, inclusive. (timestamp, e.g. "2 001-01-08T00:00:00Z")
  - groupSeqNum [optional] - only list the document in this group assigned this sequence number. (integer, e.g. 9188)
  - sinceSeqNum [optional] - only list documents in this group with a sequence number greater than or equal to this value. (integer, e.g. 9100)
- maxItems [optional, default=all] - the returned list should not exceed this number of items. (integer, e.g. 50)

#### 4.4.3.8 ListDocsRsp+

Carries the response to the request.

It has the following parameters:

- docInfoList - a list of items, which describe the documents meeting the criteria specified on the request. Each item consists of three subfields:
  - docId - the universally unique identifier (UUID) for this IPDRDoc. (UUID, e.g. 2fac1234-31f8-11b4-a222-08002b34c003. See [1])
  - docTime - the time this document was created. (timestamp, e.g. "2001 -01-08T00:00:00Z")
  - groupSeqNum - the sequence number associated with this document in the group. (integer, e.g. 9188)

#### 4.4.3.9 ListDocsRsp- (see general NegativeRsp)

Rejects a document list request.

#### 4.4.3.10 SubscribeReq

A request for the BSS to acquire a subscription to a group (either document push or demand poll).

It has the following parameters:

- version - the version of the NDM-U protocol in use, and expected for responses. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. In the context of a subscription request, this contains the necessary information for the IPDR Transmitter to initiate messages back to the requestor. (string, URL identifying the requestor, e.g. "http://voip\_bss.example.com:6615/bss1")
- groupId - the group, which the requestor wants to receive, push indications of subsequent IPDR documents. (string, e.g. "voip1\_group")
- beginSeqNum [optional, default=0] - the earliest sequence number from which to begin transmitting documents. If the IPDR document of this sequence number exists in the group, then this document and all subsequent documents (ordered by sequence number) in the group will be transmitted. If this document has already been aged out within the group, then the lowest sequence number available which is greater than beginSeqNum should begin the stream. (integer, e.g. 7000)
- idOnly [optional, default="N" ] - a Boolean flag indicating whether the requestor wants the documents transmitted on the PushReq, or just the documents identifier (the latter case corresponds to the "Demand Poll" model v s. true Push). (boolean, e.g. "Y")

#### 4.4.3.11 SubscribeRsp+

Carries the response to the request.

It has the following parameters:

- groupId - the group actually subscribed to. For this version, this value should match that requested. (string, e.g. "voip1\_group")
- beginSeqNum - the actual sequence number within the group from which PushReq's will begin. (integer, e.g. 7132)

#### 4.4.3.12 SubscribeRsp- (see general NegativeRsp)

Rejects a subscription request.

#### 4.4.3.13 UnsubscribeReq

A request for the BSS to release a subscription to a group.

It has the following parameters:

- version - the version of the NDM-U protocol in use, and expected for responses. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. This is necessary to identify an existing subscription. (string, URL identifying the requestor, e.g. "http://voip\_bss.example.com:6615/bss1")
- groupId - the group that the requestor wants to unsubscribe from. (string, e.g. "voip1\_group")

#### 4.4.3.14 UnsubscribeRsp+

Carries the response to the request.

It has no parameters.

#### 4.4.3.15 UnsubscribeRsp- (see general NegativeRsp)

Rejects an unsubscribe request.

#### 4.4.3.16 PushReq

A request sent by the IT to the BSS when the next document is available or it wakes up to retransmit, an unacknowledged document.

It has the following parameters:

- version - the version of the NDM-U protocol in use, and expected for responses. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. Note that in this case the requestor is the IPDR Transmitter in all other cases it is the BSS end of the D interface. (string, URL identifying the requestor, e.g. "http://voip\_it1.example.com:6614/voip\_it")
- groupId - the subscribed group that this document is associated with. (string, e.g. "voip1\_group")
- docId - the universally unique identifier (UUID) for this IPDRDoc. (UUID, e.g. 2fac1234-31f8-11b4-a222-08002b34c003. See [1])
- groupSeqNum - the sequence number associated with this document in the group. (integer, e.g. 9188)
- IPDRDoc [optional, dependent on the "idOnly" flag setting at time of subscription] - the information content of the IPDRDocument. (IPDRDoc, this may be either the compact encoding or XML encoded document.)

#### 4.4.3.17 PushRsp+

Carries the response to the request.

It has no parameters.

#### 4.4.3.18 PushRsp- (see general NegativeRsp)

Rejects a document push request.

#### 4.4.3.19 PullReq

A request sent by the BSS to the IT to retrieve a specific document.

It has the following parameters:

- version - the version of the NDM-U protocol in use, and expected for responses. (string, e.g. "3.0")
- requestorId - an identifier of the requestor. This is useful for log and audit purposes today and BASIC authorization. (string, URL identifying the requestor, e.g. "http://voip\_bss.example.com:6615/bss1")
- groupId - the group that this document is associated with. (string, e.g. "voip1\_group")
- [exactly one of the following two is required]
  - docId - the universally unique identifier (UUID) for this IPDRDoc. (UUID, e.g. 2fac1234-31f8-11b4-a222-08002b34c003. See [1])
  - groupSeqNum - the sequence number associated with this document in the group. (integer, e.g. 9188)

#### 4.4.3.20 PullRsp+

Carries the response to the request.

It has the following parameters:

- groupId - the group that this document is associated with. (string, e.g. "voip1\_group")
- groupSeqNum [optional, default none] - the sequence number associated with this document in the group. (integer, e.g. 9188)
- docId - the universally unique identifier (UUID) for this IPDRDoc. (UUID, e.g. 2fac1234-31f8-11b4-a222-08002b34c003. See [5])
- IPDRDoc - the information content of the IPDRDocument. (IPDRDoc, this may be either the compact or XML encoded document.)

#### 4.4.3.21 PullRsp- (see general NegativeRsp)

Rejects a request to retrieve (pull) a document.

#### 4.4.3.22 NegativeRsp

The negative response is a single primitive which primitive that is used to indicate a failure on any request.

It has the following parameters:

- reasonCode - a numeric identifier indicating the recipient of a request failed to process it. (integer, e.g. 3)  
*Current code set:*
  - 1 - no such version. The recipient does not support the version requested.
  - 2 - no such primitive. The recipient accepts the proposed version, but does not support this particular primitive.
  - 3 - unauthorized. The requestor is not recognized as having permission to perform this request.

- 
- 4 - no such group. A group specified on a request does not exist.
  - 5 - groupSeqNum not yet available. No documents in this group have been assigned this high of a sequence number.
  - 6 - groupSeqNum aged off. The document in this group is no longer available to this transmitter.
  - 7 - changeSeqNum. The receiver would like a different sequence number pushed.
  - 8 - docId is not available. The requested document is not available in this group (either because the document Id was never available or because it has been aged off)
  - 9 - already subscribed. A subscription request has already been fulfilled for this requestor and group.
  - 10 - already unsubscribed. An unsubscribe request has arrived for a requestor that is not recognized as subscribed.
- delayHint [optional, default none] - hint indicating a recommended amount of time in milliseconds for the requestor to wait before issuing a similar request. The requestor may ignore the hint if present. (integer, e.g. 60000)
  - seqNumHint [optional, default none] - hint indicating the closest sequence number, which can be delivered or is expected. A requestor may use this information to alter the next document in a group pushed or pulled.
  - versionHint [optional, default none] - indicates a preferred version, which may not result in errors. (string, e.g. "3.0")
  - primitiveHint [optional, default none] - indicates the set of primitives, which are supported by the receiver of a request. A single name identifies the request and response primitives that are supported by this IPDRTransmitter. (string list, e.g. "Capability, ListGroups, Subscribe, Unsubscribe, Push")

### 4.4.4 State Diagrams

The following state diagrams depict the states and transitions used by the NDM-U Transfer Protocol.

For each transfer scenario, there is a state diagram associated with the IT and another associated with the BSS.

The state diagram attempts to provide a protocol neutral way to depict the behavior of the IT and BSS during exchanges of IPDR documents.

#### 4.4.4.1 Reading the State Diagrams

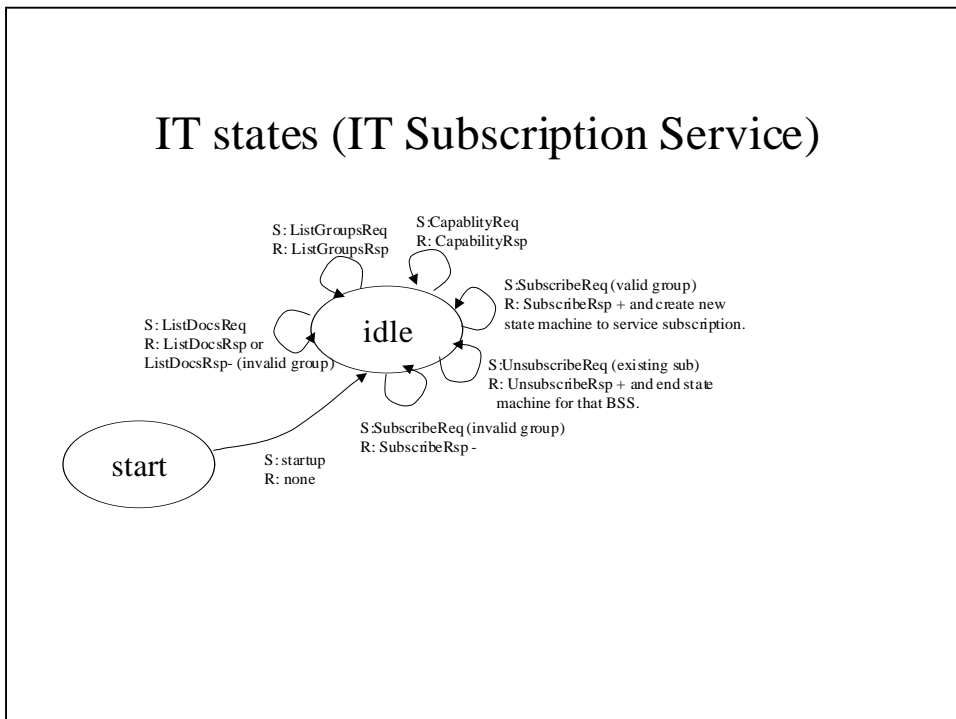
Each diagram depicts a set of states and the transition arcs between states are labeled with the Stimulus ("S") which causes a state transition and the Response ("R") to the stimulus.

All of the state machines contain the start state. Most state machines do not define a terminal state, as normal operation of these entities is to continue indefinitely. The behavior on system shutdown and restart is for the state machine to begin again at the start state. It is assumed that necessary state information is maintained across system restart. This includes last sequence number sent, information about subscriptions, etc.

Some of the stimulus or response items in the diagram also describe a condition. This is represented by a parenthesis after the name of the stimulus or response event. For example "S: PushReq (gap)", indicates a push request arrived but it is not the expected next document.

##### 4.2.4.2 Subscription Services (IT)

The IPDR Transmitter supports various inquiries about its capabilities and documents. These are lumped together as the Subscription Services. The diagram shows that upon start up any inbound request arriving at the IT is simply serviced by the appropriate positive or negative response.

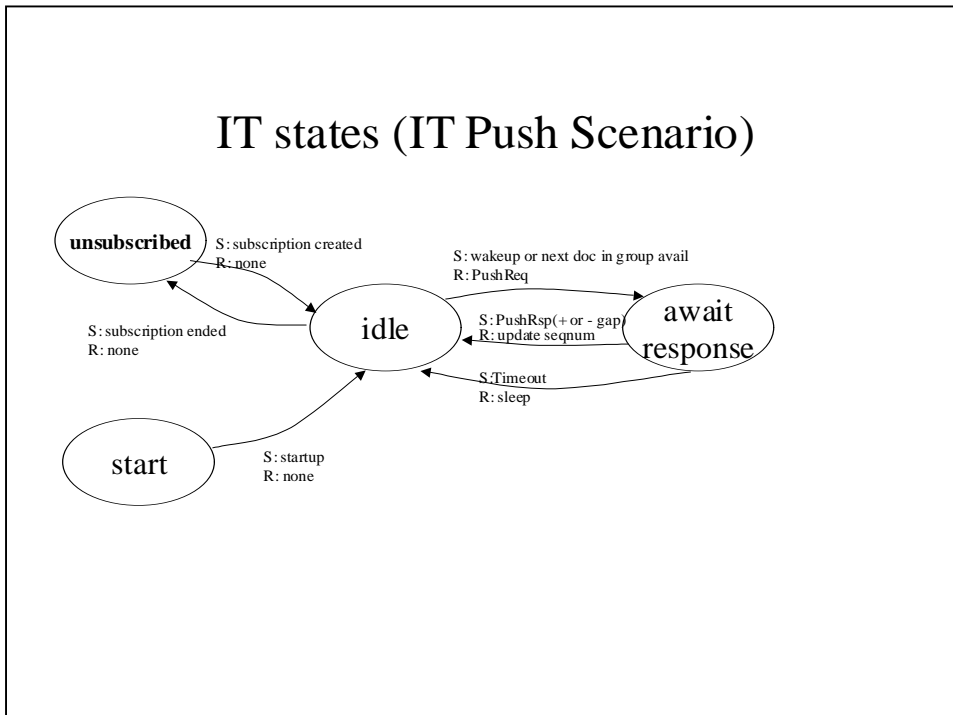


#### 4.4.4.2 Push Service (IT)

The IPDR Transmitter manages a separate state machine for each active subscription. As new documents arrive in the group, the transmitter attempts to deliver them to the BSS via PushReq messages. The transmitter will typically await a response before proceeding. If a response is not received in a configured amount of time  $t_1$ , the sender will pause for a configured interval of time  $t_2$  and then attempt to deliver the same document. This implies that a receiver may receive a duplicate IPDRDoc after a communication failure and recovery. The docId, or groupId and groupSeqNum fields allow a BSS to identify duplicates.

Moving to the unsubscribed state effectively ends this instance of the state machine.

Note that parallel transmission of documents between an IT and BSS system could be conducted across multiple communication channels.

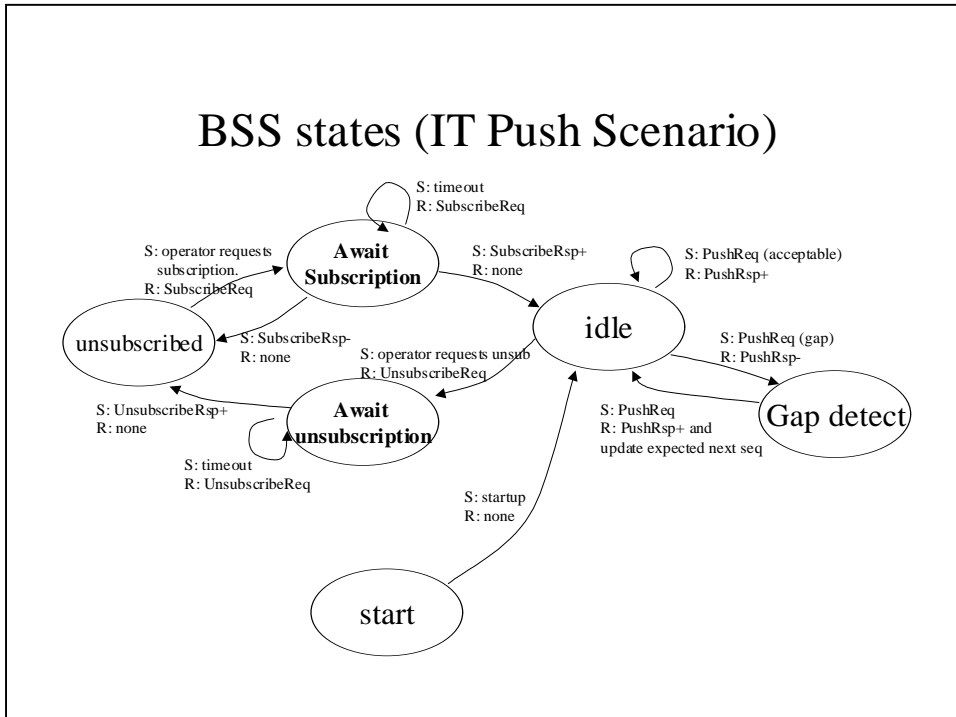


#### 4.4.4.3 Push Service (BSS)

A subscribing BSS system must first complete a subscription to an IT and then will receive a stream of documents. The documents contain a sequence number associated with the group identifier allowing gaps or duplicates to be detected in the stream.

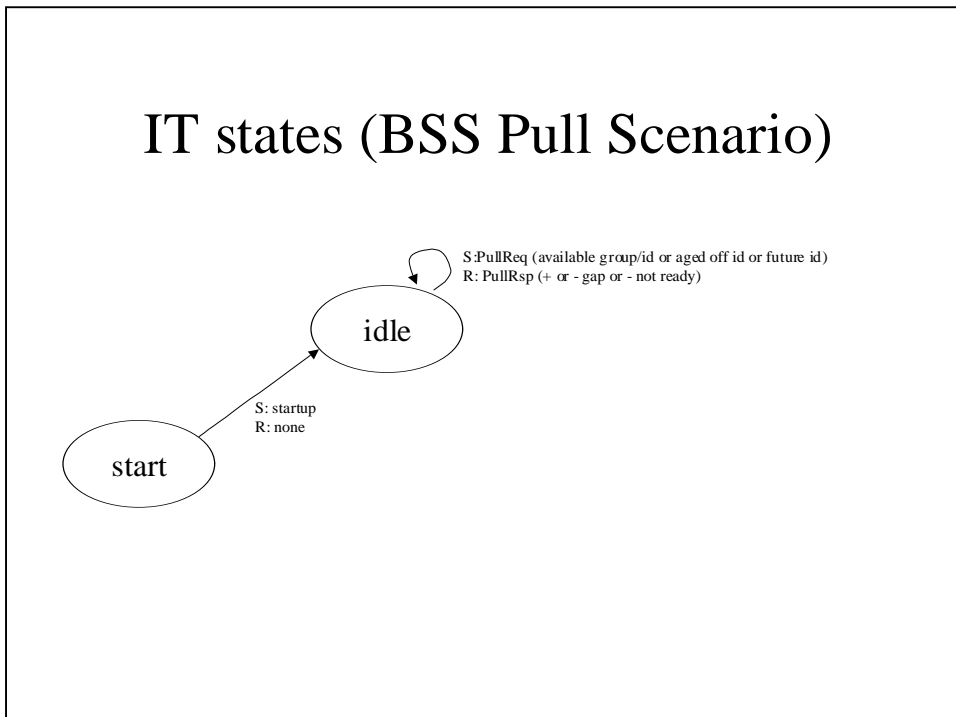
Moving to the unsubscribed state effectively ends this instance of the state machine.

Note that parallel transmission of documents between an IT and BSS system could be conducted across multiple communication channels.



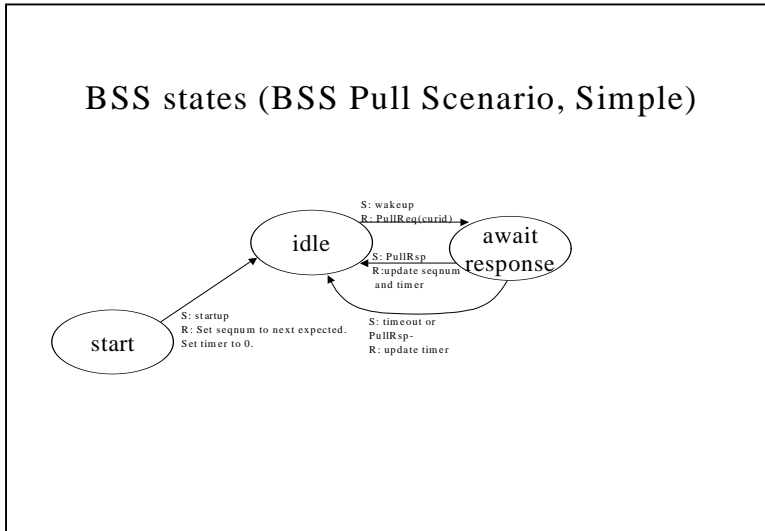
#### 4.4.4.4 Pull Service (IT)

The pull service is effectively stateless from the IT's perspective. Documents are requested in the context of a group either by DocId or groupSeqNum. If available they are delivered, otherwise an error is sent.



### 4.4.4.5 Pull Service (BSS)

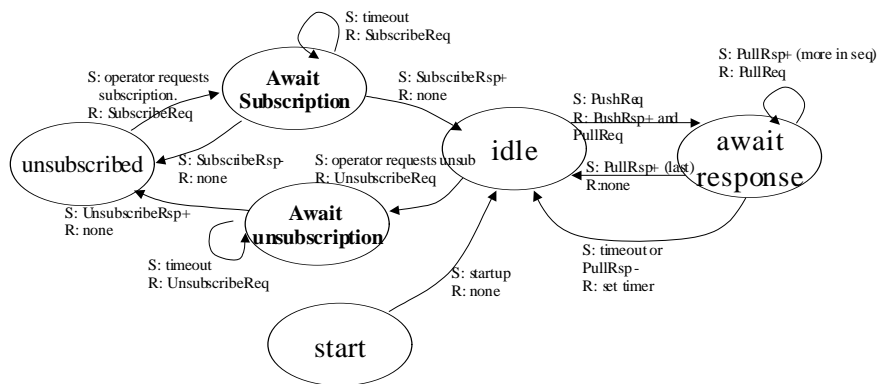
The BSS can continue to pull documents by incrementing the groupSeqNum parameter. When a response indicates the document is not available, the BSS will wait and then repoll for events. An alternative to timer based polling, demand polling may be used.



### 4.4.4.6 DemandPoll Service (BSS)

The BSS can maintain a subscription to a group, but chose to only receive the docId and groupSeqNum, not the document itself. These notifications can then replace the need for timers.

### BSS states (BSS Pull Scenario, Demand Poll)



## 4.4.5 State Definitions

### 4.4.5.1 Unsubscribed

For the Push and Demand Poll dialogs there is a state associated with being "unsubscribed". BSS systems subscribe to receive events from the IPDR Transmitter. Once subscribed the state machines will begin from the "start" state, but prior to subscription, the system is considered unsubscribed.

### 4.4.5.2 Start

This is the starting point for each state machine. When initialization of the system is complete, represented by the stimulus "startup", this state is left. Systems only return to the Start state by some externally defined restart mechanism.

### 4.2.5.3 Idle

A system is in this state when there is no active communication with its peer.

### 4.2.5.4 Await Response

A system is in this state when an IPDR document has been transmitted or requested, but acknowledgement from the peer has not yet been received.

### 4.2.5.5 Await Subscription

A BSS will enter this state when subscribing to a group.

### 4.2.5.6 Await Unsubscription

A BSS will enter this state when unsubscribing from a group.

### 4.2.5.7 Gap Detect

This state is entered by a BSS system when it receives an IPDRDoc from the IT that it does not expect.. This can happen when a BSS is down for a prolonged period of time and aging policies on the IT side result in an IPDR document being removed from the IT system.

The methods for addressing this situation are considered a policy of the system. The protocol provides some simple means to attempt resynchronization. A BSS System always has the pull mechanism available to retrieve documents in an "out of band" manner.

A BSS system may leave the Data Gap state either based on automated policy or operator intervention.

#### 4.4.6 Stimulus and Responses

The majority of the stimulus and responses in the system are messages: push or pull requests or responses or administrative requests or responses.

In addition to the protocol messages, the systems maintain timers in states containing the word "Await". Timer expiration is the second most common stimulus.

The duration of the timers is an implementation matter and is not defined by this protocol.

The state machine also has some decisions keyed off the IPDR Document sequence number specified. The sequence numbers allow the IT and BSS to maintain synchronization of the documents transferred.

#### 4.4.7 Capability Files and Protocol Extension

To address anticipated work by the IPDR Organization's Protocol Working Group in the area of both refining the protocol and creating additional mappings to address compactness and processing efficiency, a general extension model is defined.

This model allows a BSS to identify the capabilities available from a given IPDR Transmitter. In particular it defines a simple "bootstrapping" model to identify all future capabilities.

All IT's should have associated with them a "Capability URL". This URL identifies a file, which contains the XML encoding of a CapabilityRsp primitive (see Primitives and Parameters section).

For example an IPDR Transmitter may have the Capability URL of [http://voip1.myco.com/voip1\\_capabilities.xml](http://voip1.myco.com/voip1_capabilities.xml). The specific file naming is determined by the IT itself, but it is recommended to end with the ".xml" suffix.

This would imply through the default HTTP port, there should be an XML document describing the capabilities of that IT.

By parsing this XML document an implementation can identify instances of all versions it recognizes, supported by the IT. In addition for those recognized protocol versions and mappings, additional extension information can be parsed.

4.2.8 The capabilities file documents the particulars of the protocol implementation for the IT with which it is associated. The following is the schema of the capabilities file:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/2000/10/XMLSchema-->
<xsd:schema xmlns = "http://www.ipdr.org/namespaces/ipdrCap"
  targetNamespace = "http://www.ipdr.org/namespaces/ipdrCap"
  xmlns:xsd = "http://www.w3.org/2000/10/XMLSchema"
  version = "3.0"
  elementFormDefault = "qualified"
  attributeFormDefault = "unqualified">
  <xsd:complexType name = "extensionType">
    <xsd:sequence minOccurs = "0" maxOccurs = "unbounded">
      <xsd:any namespace = "##any" processContents = "lax"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name = "supportedProtocolItemType">
    <xsd:sequence>
```

```

        <xsd:element ref = "primitiveList"/>
        <xsd:element ref = "extension"/>
    </xsd:sequence>
    <xsd:attribute name = "version" use = "required" type = "xsd:decimal"/>
    <xsd:attribute name = "protocolMapping" use = "required">
        <xsd:simpleType>
            <xsd:restriction base = "xsd:NMTOKEN">
                <xsd:enumeration value = "File"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name = "encoding" use = "required">
        <xsd:simpleType>
            <xsd:restriction base = "xsd:NMTOKEN">
                <xsd:enumeration value = "XML"/>
                <xsd:enumeration value = "XDR"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
<xsd:complexType name = "supportedProtocolListType">
    <xsd:sequence>
        <xsd:element name = "supportedProtocolItem" type =
            "supportedProtocolItemType" maxOccurs = "unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name = "CapabilityRsp">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name = "supportedProtocolList" type =
                "supportedProtocolListType"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name = "primitiveList">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name = "primitiveItem" type = "xsd:string" maxOccurs =
                "unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name = "groupInfoList">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name = "groupInfoItem" type = "groupInfoItemType"
                maxOccurs = "unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:complexType name = "groupInfoItemType">
    <xsd:sequence>
        <xsd:element ref = "groupId"/>
        <xsd:element ref = "controlFileDirectory"/>
        <xsd:element ref = "controlFilePrefix"/>
    </xsd:sequence>
</xsd:complexType>

```

```
        <xsd:element ref = "controlFileNamePolicy"/>
        <xsd:element ref = "controlFileSuffix"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name = "primitiveItem" type = "xsd:string"/>
<xsd:element name = "controlFileDirectory" type = "xsd:uriReference"/>
<xsd:element name = "controlFilePrefix" type = "xsd:string"/>
<xsd:element name = "controlFileNamePolicy" type = "xsd:string"/>
<xsd:element name = "controlFileSuffix" type = "xsd:string"/>
<xsd:element name = "groupId" type = "xsd:string"/>
<xsd:element name = "extension" type = "extensionType"/>
<xsd:element name = "transmitterId" type = "xsd:uriReference"/>
</xsd:schema>
```

---

## 4.4.8 File Sharing Protocol Mappings

### 4.4.8.1 Rationale

This is a file sharing implementation of the transfer protocol. It supports minimal demand polling. IPDR.org has hosted several interoperability demonstrations based on this file-oriented scheme. It is relatively easy to implement, but lacks the rich feature-set supported by the full state machine specification of the transfer protocol.

### 4.4.8.2 Overview

This section describes the basic terminology and mechanics of the file sharing implementation of the transfer protocol, while sections 4.2.10.3 thru 4.2.10.7 provide a more thorough description of the file sharing transfer protocol.

The abstract transfer protocol, described in sections 4.2.1 thru 4.2.7, is based on the assumption that there is request/response based communication channel between an IPDR Transmitter (for the purposes of this discussion this will be referred to as the IPDR Producer) and a BSS (henceforth referred to as an IPDR Consumer). Since this file sharing implementation of that transfer protocol has no such formal communication channel, many of the details presented in the transfer protocol section won't map to this file sharing scheme in an explicit way. However many key concepts do map from the transfer protocol to this file sharing approach. Section 4.2.10.2.6 describes how the primitives defined in the general transfer protocol map to the elements of the file sharing implementation of the transfer protocol.

#### 4.4.8.2.1 IPDRDocs and Subscription Groups

As an IPDR Producer creates IPDR documents (abbreviated as IPDRDocs) it places them into ASCII files, with one IPDRDoc per file. The IPDR Producer sorts each IPDRDoc into one or more groups. IPDRDocs will often be grouped by service type, by customer name, or by a combination of the two, though the IPDR Producer is also allowed to use any other useful policy to sort IPDRDocs into groups. These groups are referred to as subscription groups to match the terminology used in the transfer protocol. Subscription groups provide an easy way for an IPDR Consumer to access only those subsets of IPDRDocs that are of interest to it.

#### 4.4.8.2.2 Control files and Control file Rolling

As the IPDR Producer places an IPDRDoc into a subscription group, it adds the name (and possibly the location) of the IPDR document file to the last line of the current control file for that subscription group. The names of the control files follow a predictable naming convention, which is <prefix><sequenceNumber><suffix>. The IPDR Producer may periodically create a new control file and name it by incrementing the current sequence number by one. This process of periodically switching to a new control file is referred to as "rolling". When the Producer does this, it places a special end-of-file string at the end of the previous control file, so that any consumers who are reading from subscription group will know to stop reading the old control file and to open the new control file. A common rolling policy is one that uses date-time stamps, such that the IPDR Producer would change control files every hour or every day. However the IPDR Producer is free to implement any "rolling" policy.

#### 4.4.8.2.3 Capability File

The IPDR Producer maintains a capability file, which has a list of all the subscription groups. The capability file contains all the information an IPDR Consumer needs to “subscribe” to one or more subscription groups. In this file sharing implementation, subscribing simply involves the IPDR Consumer copying or ftping the IPDRDocs from a subscription-group. In this file-sharing scheme, the IPDR Producer has no knowledge of which IPDR Consumers are reading its IPDRDocs, nor does it know which IPDRDocs have been accessed.

#### 4.4.8.2.4 Range File

All of the control files for a specific subscription group are contained in one directory and the path to this directory is in the capability file. The control file directory also contains a range file for that subscription group. It may also contain the IPDR document files to which the control files refer but this is not required. The range file is used IPDR Consumers subscribed to the subscription-group to determine which control files currently exist for that subscription group, and most importantly which control file is currently having the names of new IPDR document files appended to it. The range file contains two sequence numbers separated by a dash. The first is the sequence number of the oldest control file. The second is the fixed-length sequence number of the current control file. At times, if the size of the sequence number isn't large enough to handle the number of control files, the sequence number may reach its maximum value and then roll back to all zeroes. So it is possible that the left-hand sequence number in the range file may be larger than the right-hand sequence number. An IPDR Consumer that has just subscribed to a subscription group would read the range-file for that group to get the sequence number of the oldest control-file. Using that sequence number, the IPDR Consumer could determine the name of the oldest control-file and start reading the IPDRDocs to which that file refers. By consecutively incrementing the sequence number, it could then read through all control-files in order till it reached the end of the most recent one

#### 4.4.8.2.5 Aging of IPDR Documents and Control files

Multiple control files are used to make it easy for the IPDR Producer to periodically remove old IPDRDoc, a process known as “aging”. Using the range file the IPDR Producer can easily identify the oldest control files and remove those control files as well as the IPDRDocs to which those control files refer. When removing old IPDR documents and control files, the range file would also be updated to reflect the recent deletions. The IPDR Producer is free to implement any aging policy. One common aging policy would involve checking the control files on a daily basis, and deleting all control files that are more than a week old. However, in high volume environments the check might be done on an hourly basis and the IPDR Producer might remove all files that are more than a day old. The file sharing implementation of the transfer protocol provides an infrastructure that supports a variety of aging policies, so the choice is up to the IPDR Producer.

#### 4.4.8.2.6 Mappings from Transfer Protocol to File Sharing Implementation

There are no explicit primitives for CapabilityReq, ListGroupsReq, and ListGroupsRsp; however the functions performed by these primitives can still be accomplished within the file sharing implementation. Instead of issuing a ListGroupsReq and waiting for the ListGroupsRsp, the IPDR Consumer can get a list of subscription groups by reading the capability file and reading the <groupInfoList> and <groupInfoItem> elements. Rather than exchanging ListDocsReq and ListDocsRsp messages to get a list of all the

documents in a subscription group, the IPDR Consumer reads the range file and control files for that subscription group. The IPDR Consumer subscribes to a subscription group simply by monitoring available files without the explicit knowledge of the Producer, so there are no analogues in the file-sharing mapping to the SubscribeReq, SubscribeRsp, UnsubscribeReq, UnsubscribeRsp, PushReq, or PushRsp primitives. The IPDR Consumer accomplishes a PullReq/PullRsp simply by copying or ftping an IPDRDoc referred to in a control file for a particular subscription group.

### 4.4.8.3 IPDR Document Files

Each IPDR document must be placed in an ASCII file. Each file must contain exactly one IPDRDoc. These IPDR document files are not required to follow any specific naming convention, however, each IPDR document file name must be unique. The IPDR Producer is responsible for ensuring that no two IPDR documents are created with identical names. It may be useful to incorporate such information as the IPDR Producer name, a date/time stamp, and/or the IPDRDoc UUID into the file name.

### 4.4.8.4 Subscription-group Control-files

Each subscription group has one or more control files. Each control file is an ASCII file that lists a subset of file names corresponding to the IPDR documents included in that subscription group. The first line of each control file must always be "VERSION 2" followed by a linefeed character.

Each subsequent line of the control file contains either the URL or the file name of exactly one IPDR document file. If just the IPDR document file name is used, it is assumed to reside in the same directory as the control file that references it. Each line of text in the control file should be terminated with exactly one linefeed.

The IPDR Producer must append a "VERSION 2" to the end of the control file to denote that no more IPDR document filenames will be added to that control file.

The maximum number of lines in a control file is unconstrained. The minimum number of lines in a control file is 1 while the control file is still "active" (ie. the IPDR Producer is still adding IPDR document filenames to that control file), and 2 once the control file is no longer having new IPDR document filenames added to it. A minimal control file, that is no longer active would look like this (where <EOF> denotes the end-of-file):

```
VERSION 2
VERSION 2
<EOF>
```

The name of the control file is required to follow the naming convention specified in the IPDR Producers capability file. The control file name has unchanging prefix and suffix, which are defined in the capability file using the <controlFilePrefix> and <controlFileSuffix> elements. In between these there must be a sequence number. The sequence number is a decimal integer with the number of digits specified by the <controlFileNamePolicy> in the capability file. The sequence number part of the control file name must always have the number of digits specified in the capability file and must use leading 0's to effect this.

The presence of an IPDR document filename or URL in the control file implies that the named IPDR Document file must exist. The IPDR Producer is responsible for ensuring that files cited in the control files are actually available at the referenced location.

The IPDR Producer is responsible for ensuring that control file references point only to valid, fully formed, completed (no longer changing) IPDR documents. This implies that the IPDR Producer should finish writing to an IPDR document file before appending that file's name to the control file, and should delete control files before deleting IPDR documents.

When reading the control file, the IPDR Consumer is required to open it in read-only mode. When adding new IPDR documents to a subscription-group, the IPDR Producer should open the control file in append mode.

#### 4.4.8.5 Control File Rolling and Aging

In production environments, a single control file is insufficient because over time it will grow very large. In addition this would imply that the referenced IPDR document files are kept indefinitely (since they are referenced by the control file).

An optional rolling policy can be employed to indicate that a new control file will be created periodically. The use of the rolling option requires that control file names contain a sequence number embedded between the prefix and the suffix. The number of digits in that sequence number shall be held constant and the numeric value shall rise incrementally (recycling to 0 upon hitting the highest value).

If the <controlFilePrefix> was "voip\_IT1\_" , the <controlFileNamePolicy> was "NNNNNN", and the <controlFileSuffix> was ".log", then the control file names would be generated in the following order:

```
voip1_IT1_000000.log
voip1_IT1_000001.log
voip1_IT1_000002.log
...
voip1_IT1_999997.log
voip1_IT1_999998.log
voip1_IT1_999999.log
voip1_IT1_000000.log
voip1_IT1_000001.log
...
```

Note how the sequence number wraps around from 999999 to 000000. This is the only time that the sequence numbers won't be monotonically increasing. It is the responsibility of the IPDR Consumers to sequence number rollover in the particular case where the range file has a left-hand sequence number that is larger than the right-hand sequence number. IPDR Producers are strongly encouraged to choose a sufficient number of digits for the sequence number to prevent rollover from occurring within the time interval in which this file sharing transfer protocol will be used in a production environment.

If the rolling policy is used, the last line of a completed control file shall always be "VERSION 2". This implies that no additional appends will be made to this control file and a higher numbered control file is available; the IT is responsible for ensuring this to be true. Only one control file per group may be open for appending by the IT.

The IPDR Producer may delete old control files and corresponding IPDR document files based on locally defined aging policies, however the list of existing control files must always remain contiguous. For instance, it would be incorrect for an IPDR Producers to delete control file voip-0004.log if both voip-0003.log and voip-0005.log still exist, since that would introduce a discontinuity in sequence numbers.

#### 4.4.8.6 Capability File Annotated Description

The capability set documented here corresponds to the following entry in the capability file:

```
<supportedProtocolItem version=3.0 protocolMapping="File" encoding="XML">
  <primitiveList>
```

```

    <primitiveItem>Pull</primitiveItem>
  </primitiveList>
  <extension>
    <groupInfoList>
      <groupInfoItem>
        </groupInfoItem>
      </groupInfoList>
    </extension>
  </supportedProtocolItem>

```

Each subscription group must have a `groupInfoItem` element containing the following sub-elements:

```

<groupId></groupId>
<controlFileDirectory></controlFileDirectory>
<controlFilePrefix></controlFilePrefix>
<controlFileNamePolicy></controlFileNamePolicy>
<controlFileSuffix></controlFileSuffix>

```

All of these elements are constrained to contain a string containing only the following characters: '0 -9'a -z,' A-Z',' ;' -;',' /, a nd ' \_'.

#### 4.4.8.6.1 <groupId>

The `<groupId>` element must contain the name of the subscription group described by this `<groupInfoItem>` element. It should be a non-zero length string of allowable characters. Some examples might include: "voip", "e-mail-1", and "best-hosting-co".

#### 4.4.8.6.2 <controlFileDirectory>

The `<controlFileDirectory>` contains the URL that points to the directory containing the control files and the range file for the subscription group. A URL is required here to avoid the complications that can arise when local O/S file naming conventions are used. For instance, Microsoft Windows and Unix/Linux operating systems use different characters to denote path's (Windows uses '\', while Unix uses '/'). The URL can either be a "file:" or an "http:". Some examples might include: "<file:///C:/IPDR/voip/>", "<http://www.best.com/voip/>".

#### 4.4.8.6.3 <controlFilePrefix>

The `<controlFilePrefix>` element contains a string of allowable characters that denotes the first part of the name of all of the control files in the subscription groups. Some possible examples include: "voip-", "internet-access\_", and "best.com."

#### 4.4.8.6.4 <controlFileNamePolicy>

The `<controlFileNamePolicy>` element must contain one or more 'N' characters, where the number of 'N' characters denotes the number of digits in the sequence number embedded in the name of all control files for this subscription group. Some examples include: "N NNNNN", and " NNNNNNNNNN".

#### 4.4.8.6.5 <controlFileSuffix>

The <controlFileSuffix> element contains a string of allowable characters that denotes the last part of the name of all of the control files in the subscription groups. Some possible examples include: “.log”, “-control” and “”.

#### 4.4.8.6.6 Sample Capability File

```
<supportedProtocolItem version=3.0 protocolMapping="File" encoding="XML">
  <primitiveList>
    <primitiveItem>Pull</primitiveItem>
  </primitiveList>
  <extension>
    <groupInfoList>
      <groupInfoItem>
        <groupId>sm</groupId>
        <controlFileDirectory> http://www.best.com/sm</controlFileDirectory>
        <controlFilePrefix>sm</controlFilePrefix>
        <controlFileNamePolicy>NNNNNNNN</controlFileNamePolicy>
        <controlFileSuffix>-control</controlFileSuffix>
      </groupInfoItem>
      <groupInfoItem>
        <groupId>voip</groupId>
        <controlFileDirectory>file://C/IPDR/voip/</controlFileDirectory>
        <controlFilePrefix>voip_IT1_</controlFilePrefix>
        <controlFileNamePolicy>NNNNNN</controlFileNamePolicy>
        <controlFileSuffix>.log</controlFileSuffix>
      </groupInfoItem>
    </groupInfoList>
  </extension>
</supportedProtocolItem>
```

This capabilities file describes two subscription groups: vod and voip.

The voip subscription group has all its control files located in “<file://C/IPDR/voip/>”. The names of the control files would all look like voip\_IT1\_000001.log. The following name would not be allowed voip\_IT1\_1.log since the control file name is required to have an eight digit sequence number embedded between the prefix and the suffix.

The sm (sm stands for Streaming Media) subscription group has all its control files located at “<http://www.best.com/sm>”. The names of the control-files would all look like sm-0000001-control.

#### 4.4.8.7 Range file

Each subscription group must have a range file. The range file is used by both the IPDR Producer and any IPDR Consumers subscribed to the subscription group to determine which control files currently exist for that subscription group, and most importantly which control file is currently having the names of new IPDR document files appended to it. The range file contains two sequence numbers separated by a dash. The first is the sequence number of the oldest control file. The second is the sequence number of the current control file. The name of the range file must be the name of the subscription group, found in the <groupId> element, concatenated with the string “-range-file”. The range file for a particular subscription group must reside in the same directory as the control files for that subscription group. This directory is specified by the <controlFileDirectory> element for that subscription group in the capability file.

#### **4.4.9 Security Considerations**

There is one mapping of the transfer protocol for NDM-U 3.0: File based. The File based mapping is dependent on the underlying security model maintained by the file system (or network shared file system). In a user/group model, group level permission could be associated with specific document groups. This protocol does not currently provide any details related to the usage of file system permissions.

#### **4.4.10 Implementation Considerations**

IPDR.org is currently developing reference libraries implementing the various aspects of the protocol defined in this specification. Availability of these APIs to IPDR member companies is expected in the spring of 2002.